UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERíA DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

Adquisición y gestión de *bookmarks* de delicious en comunidades .LRN

Autor: Carlos Valencia López
Tutor: Abelardo Pardo Sánchez

25 de octubre de 2009

Agradecimientos

Este proyecto es el punto y final de la carrera de Ingeniería de Telecomunicación que comencé en 2003. Por ello quiero dejar constancia del agradecimiento que tengo a todos aquellos que fueron importantes durante estos años.

En primer lugar quiero dar las gracias a mis padres, José Antonio y María del Carmen, por darme la oportunidad de estudiar una carrera y apoyarme. También quiero dar las gracias a mi familia por su apoyo así como por mostar interés por lo que hacía.

Gracias a mi tutor, Abelardo Pardo, por darme la oportunidad de realizar este proyecto así como por ofrecerme la oportunidad de comenzar mi carrera profesional a raíz de los conocimientos adquiridos durante el desarrollo del proyecto.

También quisiera dar las gracias a todos mis amigos por estar siempre conmigo. Primero quiero recordar a mis amigos del colegio, Óscar Magro, Álvaro González, Roberto Peral y Germán Mozo. También quiero agradecir el apoyo y compañía de aquellos que he conocido en la universidad: Laura Vallejo, Ana de Lucas, Antonio Palma, Alex Zurita, Alicia Rodríguez, Atanas Plamenov, Carlos Cerrón, Carlos Moriana, David Redondo, Diego Morillo, Juan González, Félix Rodríguez, Gonzalo Díaz-Tendero, Irene Sainz, Isabel Durán, María Rueda, Moisés Sánchez, Nieves Vallejo, Laura Lucas, Laura Romero, Lidia Seldas, Lucía Plaza, Pablo González-Ripoll, Patricia Lozano, Patxi Azpiroz, Roberto Pizarro, Rubén Gálvez, Sonia Collada, Valle Nieto y Víctor Sánchez.

También doy las gracias a mis compañeros de laboratorio: Álvaro, Aurora, David, Derick, Estrella, Iago, Israel, Miguel, Laura, Paola, Patri A., Patri N., Rosa y Sara.

Por último, también quiero dar las gracias al resto de gente que no he nombrado aquí pero que han estado junto a mi en algún momento de la carrera.

Resumen

En esta memoria se explica la aplicación desarrollada para la gestión de *bookmarks* en comunidades de OpenACS/.LRN. Para ello, primero se ofrece una visión del estado del arte incluyendo la plataforma OpenACS/.LRN y el lenguaje de programación TCL, así como la situación actual en cuanto a gestión de *bookmarks* se refiere que podemos encontrar en el mercado.

Se detalla el diseño del servicio, incluyendo el ciclo de desarrollo. Después se desgrana este servicio en los dos paquetes software desarrollados en el proyecto. Uno de ellos implementa la mayor parte de la funcionalidad requerida en el diseño y otro con carácter generalista diseñado para incrementar la funcionalidad del paquete con un sistema de evaluación que puede ser usado en futuros desarrollos en OpenACS.

Por último, se explica el diseño de las pruebas realizadas, se comenta la historia del proyecto incluyendo el presupuesto del mismo y un capítulo de conclusiones y líneas de investigación futuras que surgen a partir de lo desarrollado en este proyecto.

Abstract

This memory explains the management of bookmarks in OpenACS/.LRN application developed. First, a general view of the state of art is shown including OpenACS/.LRN and TCL programming language, as well as the different services related with the bookmark management in the internet.

The design of the developed service is explained, besides the development cycle. Afterwards, the service is explained in detail by splitting it into the two packets created in the project. On of them implements most of the functionality required in the design especification and the other is more generalistic because it offers an evaluation system that is used in the application but can also be used by other applications in OpenACS.

Then, the tests used to test the application are explained. The history of the project is described including the budget and a final chapter with conclusions and future lines of investigation open from the work developed in this project are also included.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Las últimas tendencias en diseño web se enfocan hacia la creación de nuevas plataformas que utilicen servicios externos existentes que los usuarios pueden agregar creando su propio entorno. Esta misma tendencia se detecta en las plataformas de gestión del aprendizaje (LMS). La integración de servicios es cada vez más y más importante debido al valor añadido que dan a las plataformas. Por otra parte, la integración de servicios permite añadir nueva funcionalidad a plataformas existentes sin tener que realizar cambios importantes en el núcleo de la plataforma.

La plataforma de código abierto .LRN [8] ha sido desarrollada para albergar comunidades virtuales que aglutinan usuarios con intereses comunes y que comparten recursos, dispositivos, servicios etc. Este concepto de comunidad virtual puede utilizarse además para dar suporte a experiencias educativas tales como cursos online o de apoyo a cursos presenciales. Una de las ventajas de esta plataforma está en que es de código abierto (*Open Source*), lo que implica que el desarrollo sobre esta plataforma y su despliegue está al alcance de cualquiera. El ámbito de diseminación de esta herramienta consta de más de medio millón de usuarios en todo el mundo.

El desarrollo del proyecto comprende una primera fase de familiarización con la plataforma, su entorno de desarrollo, modelo interno de datos y estudio de las herramientas complementarias para su desarrollo: administrador de incidencias, controlador de versiones, depósitos de archivos, máquinas virtuales, etc.

En una segunda fase se estudiarán distintas soluciones para la asimilación de información procedente de aplicaciones externas para incluirla en un módulo a integrar en la plataforma. Dada la versatilidad de operaciones disponibles en la plataforma así como el grado de abstracción de su funcionalidad más genérica, se desarrollará un módulo que tendrá un carácter generalista para maximizar su impacto, así como una aplicación concreta que

hará uso de esta funcionalidad y que será elegida en una tercera fase. Esta aplicación ofrecerá un servicio de usuario en el ámbito de una comunidad virtual.

Como idea genérica para esta tercera fase, lo que se contempla es un módulo que realice tareas de agregación de contenidos web orientado a los intereses de la comunidad virtual en la que se empotra. Basándose en las preferencias de cada usuario que ya están almacenadas en un servicio externo, obtener una agregación de recursos específica para la comunidad. Para implementar esta funcionalidad se utilizarán servicios que ofrezcan acceso a sus datos remotos (este proyecto utilizará del.icio.us [7], una aplicación que almacena páginas de interés etiquetándolas dentro de nuestras preferencias para después poder utilizar las preferencias de otros usuarios para poder ampliar la información sobre un determinado tema).

Mediante la implementación de esta aplicación se pretende demostrar que la funcionalidad adicional es correcta, y que su instanciación concreta en una aplicación puede hacerse dentro de los parámetros razonables de desarrollo.

El proyecto se concibe con un contenido innovador. La necesidad de interconexión entre servicios se deriva de la tendencia observada en la evoluación de Internet. Los usuarios no obtienen servicios exclusivamente de una única fuente, sino que se tiende a los espacios personalizados en los que el usuario combina los datos que prefiere a su gusto. El albergar esta funcionalidad en .LRN supone desarrollar la capacidad de intercambio de datos y de integrarla en su modelo interno de datos.

Puesto que .LRN es una plataforma en constante expansión, el desarrollo de este programa podrá servir como inicio de otros muchos proyectos que incluyan los nuevos módulos creados para la realización de otros módulos, así como poder incorporarse la aplicación práctica implementada en la fase tres en cursos de grado universitario.

En esta memoria se utilizará la palabra bookmarks para referirse a las páginas de interés almacenadas, también llamadas marcadores o favoritos dependiendo del navegador o aplicación.

1.1. Objetivos

En el desarrollo de este proyecto se han fijado unos objetivos que persiguen, por un lado, el aprendizaje de una nueva plataforma que nos ofrece la posibilidad de albergar comunidades virtuales y otra serie de herramientas adicionales relacionadas con el desarrollo de software y por otra parte, los objetivos también buscan demostrar todos los conocimientos adquiridos mediante la implementación de una aplicación en la que se demuestren dichos

conocimientos.

Los objetivos del proyecto son:

1. Estudio de la plataforma dotlrn: el objetivo principal consiste en desarrollar una aplicación para esta plataforma que integre servicios externos. Para ello se debe de conocer el entorno de desarrollo y el modelo de datos de la plataforma, pero también es importante conocer como funciona la plataforma desde la visión de los administradores y usuarios.
    - Modelo interno de datos
    - Entorno de desarrollo
2. Aprendizaje de lenguaje de programación TCL: es necesario aprender a utilizar este lenguaje para poder desarrollar aplicaciones en dotlrn ya que está implementado en TCL.
3. Estudio de la integración de servicios externos: la utilización de mashups está cada vez más extendida ya que los usuarios piden aplicaciones más personalizadas y la forma más sencilla de ofrecer eso es integrar servicios externos agregándoles funcionalidad propia.
4. Realización de aplicación en la que se demuestren los conocimientos adquiridos en los puntos anteriores.
5. Estudio y utilización de herramientas auxiliares en el desarrollo: durante la vida laboral de un ingeniero, este utiliza un conjunto de herramientas que no son las fundamentales para el desarrollo de su trabajo sin embargo, le otorgan una serie de funciones que son de gran ayuda y cuya utilidad crece proporcionalmente al tamaño del proyecto.
    - Administrador de incidencias: esta herramienta de apoyo para notificar problemas es utilizada para comunicar de manera rápida y eficaz sobre errores y otras cuestiones para minimizar sus efectos.
    - Utilización de controlador de versiones SVN: los controladores de versiones se utilizan para mantener una historia del desarrollo de los proyectos de software. Son de gran utilidad ya que permiten almacenar estructuradamente los cambios y de este modo recuperar cualquier versión almacenada rápidamente para poder retomar el trabajo desde un estado estable.
    - Utilización de un depósito de archivos (Content Repository): los depósitos de archivos nos ofrecen la posibilidad de almacenar tanto el código como cualquier otra documentación desarrollada o

de utilidad durante la realización del proyecto en un lugar desde el que puede acceder cualquier miembro del proyecto de forma remota.

- Utilización de máquinas virtuales (VMware): las máquinas virtuales nos permiten trabajar con varias máquinas al mismo tiempo sin necesidad de tener varios equipos. También nos permiten utilizar sistemas operativos distintos al que tenemos instalado en la máquina física.

1.2. Contenido de la memoria

La memoria se divide en 8 capítulos más tres apéndices. A continuación se resume el contenido de cada capítulo:

En el Capítulo 1 se ha dado una visión global del trabajo realizado, mencionando los objetivos que se han buscado con el desarrollo de este proyecto.

El Capítulo 2 contiene el estado del arte. En él se expone la plataforma .LRN, la base de datos PostgreSQL y el servidor AOLserver. También se realiza un estudio de las distintas alternativas que existen actualmente en la gestión de *bookmarks* en internet.

En el Capítulo 3 se realiza una descripción de la aplicación de gestión de *bookmarks* desarrollada en este proyecto, incluyendo la organización que tienen los paquetes desarrollados y como se han internacionalizado.

En el Capítulo 4 se explica el paquete Bookmark detallando el modelo relacional de la base de datos, funcionalidad y el mecanismo implementado para la actualización de la información obtenida de delicious.

En el Capítulo 5 se explica el paquete Rating, que se ha realizado para complementar la funcionalidad de gestión de *bookmarks* con evaluación de objetos. Paquete diseñado de forma genérica para que pueda ser reutilizado en otros trabajos y proyectos.

En el Capítulo 6 se detalla la estrategia seguida para la realización de las pruebas del programa.

En el Capítulo 7 se encuentran tanto la evolución temporal del proyecto como los costes asociados al mismo.

En el Capítulo 8 se recogen las conclusiones obtenidas tras el desarrollo del proyecto, así como líneas futuras para la mejora de la aplicación de gestión de *bookmarks* y otras no relacionadas con la misma pero que tienen como base este trabajo.

El apéndice A es una guía de instalación de los paquetes para poder utilizar la aplicación, el apéndice B contiene una guía de usuario que explica lo que podemos hacer en la aplicación y en el apéndice C se recopilan todos los

enlaces de servicios web, comunidades, tutoriales y otras páginas relevantes utilizadas durante el desarrollo del proyecto.

Capítulo 2

Estado del arte

Antes de comenzar con la explicación sobre la arquitectura, componentes y servicios de esta aplicación, primero es necesario tener una visión del marco en el cuál se ha desarrollado, incluyendo las bases y el entorno sobre los que se cimenta. En este capítulo se desarrollarán todos aquellos temas técnicos necesarios para un correcto entendimiento del enfoque y complejidad del trabajo desarrollado en este proyecto.

Primero se presentará el lenguaje de programción en que están escritos los paquetes desarrollados. Después se describe la plataforma sobre la que se ha implementado el servicio, la base de datos y el servidor. Posteriormente se describen los principales gestores de *bookmarks* que existen actualmente en internet y se detalla el funcionamiento del API de delicious. Por último, se comenta que son los *mashups*.

2.1. Lenguaje de programación TCL

TCL ha sido el lenguaje con el que se ha implementado la aplicación descrita en este proyecto. Se ha utilizado TCL debido a que es el lenguaje utilizado por la plataforma sobre la que se quería implementar dicha aplicación y como se mostrará a continuación, TCL fue el lenguaje elegido por dicha plataforma debido a las ventajas que ofrecía respecto a otro lenguajes.

2.1.1. Historia

El lenguaje de programación TCL fue creado en la primavera de 1988 por John Ousterhout. Para conocer que le motivó para crear este lenguaje, se puede leer lo que el mismo dijo:

> "El lenguaje de programación TCL surgió de mi trabajo sobre herramientas de diseño de circuitos cerrados para la Universidad de California en berkeley. [...] Cada herramienta necesitaba tener un lenguaje de comandos. [...] Sin embargo, nuestro principal interés estaba en las herramientas, no en los lenguajes de comandos. Por lo tanto, haber ahorrado tiempo en no investigar demasiado sobre el lenguaje de comandos nos resultó contraproducente en el futuro."

TCL surgió como una necesidad para ahorrar tiempo ya que cada vez que usaban una nueva herramienta tenían que diseñar primero un lenguaje para la herramienta, algo que era muy ineficiente ya que no podían reutilizar dichos lenguajes.

> "En el otoño de 1987, tras un año sabático en DEC's Western Research Laboratory, tuve la idea de construir un lenguaje de comandos que fuese insertable para cualquier aplicación."

Este lenguaje sería interpretado, no compilado. Además, se diseñó el lenguaje para construirlo como un paquete de biblioteca de modo que pudiese ser reutilizado por muchas aplicaciones diferentes. El intérprete del lenguaje ofrecería una serie de unidades genéricas tales como variables, estructuras de control y procedimientos. Este lenguaje permitiría agregar nueva funcionalidad fácilmente de modo que el lenguaje fuese extensible.

> "Cada aplicación podría añadir nuevas características al lenguaje de comandos en forma de extensiones, por lo que el lenguaje podría ser utilizado para manejar la aplicación. El nombre TCL (Tool Command Language) derivó de la intención de esta finalidad.
>
> La noción de insertabilidad es uno de los aspectos más singulares de TCL. [9]"

Ousterhout tenía tres objetivos principales cuando comenzó con el desarrollo de este lenguaje de programación:

- Debía ser un lenguaje extensible, ya que era necesario que se pudieran añadir nuevas funciones y características a las básicas existentes en el lenguaje en cada nueva aplicación. Además, las aplicación de las nuevas características debía asimilarse a las ya existentes en el lenguaje.

- Debía ser un lenguaje simple y genérico para poder trabajar con muchas aplicaciones y que sus procedimientos no interfiriesen con los procedimientos que una aplicación pudiese proporcionar.

- El lenguaje debía tener buenas instalaciones para la integración ya que la mayoría de procedimientos importantes provendrían de la aplicación.

2.1.2. Características y ventajas de TCL

- Rápido desarrollo de aplicaciones:

  La razón más importante por la que se sigue utilizando TCL es que se puede completar el trabajo rápidamente. En muchos casos se puede llegar a implementar la aplicación mucho más rápido que con otros lenguajes, especialmente en aquellas aplicaciones donde se utilizan GUIs, manejo de cadenas de texto...Esto se debe a la gran cantidad de código reutilizable debido al uso de procedimientos para la implementación de tareas. Una vez desarrollada, una aplicación también puede ser rápidamente modificada para soportar nueva funcionalidad.

- Aplicaciones multiplataforma

  TCL puede ser utilizado sobre distintas plataformas como Windows, Macintosh y plataformas Unix. Además, provee un API de alto nivel que permite escribir código que puede ser utilizado de la misma forma en todas las plataformas, mientras que el interprete es el que se encarga de tratar con las diferencias entre ellas.

- Fácil de aprender

  TCL es un lenguaje simple. Puede ser utilizado tanto por programadores expertos como inexpertos y permite aunar el trabajo de ambos separando el núcleo de la aplicación para programadores expertos y la personalización de este núcleo, reglas de negocio, etc para programadores no expertos.

- Maduro y evolucionando

  TCL has estado en constante desarrollo y uso por un gran número de programadores desde principios de los 90. Esta es la razón por la que TCL tiene todos aquellos toques finales surgidos durante la implementación de aplicaciones reales. TCL es un lenguaje fiable, de alto rendimiento, portable, integrable, con un gran soporte internacional, seguro con la utilización de hilos, soporta tareas en red y otras que hacen recomendable su uso. Debido a que TCL está constantemente

en evolución, nuevas características aparecen periódicamente, siempre manteniendo la compatibilidad hacia atrás.

- Extensible, empotrable e integrable

  TCL es fácilmente empotrable como lenguaje de *script* dentro de una aplicación, o en código C, C++ o Java existente. Las mismas características hacen que TCL sea un lenguaje de fácil utilización como lenguaje de control para hardware de utilidad específica.

- Despliegue

  Los lenguajes dinámicos suelen dificultar el despliegue porque se necesita tener tanto el intérprete del lenguaje como los scripts de la aplicación en la misma máquina. La mayoría de estos lenguajes ofrecen una herramienta para compilar todo en un solo ejecutable (TCL lo tuvo hasta 1993). Pero TCL ha ido más allá de esas soluciones simples y usando tecnologías como la del sistema virtual de archivos o paquetes estrella ha conseguido tener un despliegue mucho más flexible, potente y transparente. Otras opciones permiten proteger la propiedad intelectual en aplicaciones comerciales, capacidad rara en lenguajes dinámicos.

- Pruebas

  La implementación interpretada de TCL permite crear tests rápidamente, además, TCL permite utilizar APIs de bajo nivel directamente, lo cuál permite una fase de pruebas mucho más precisa y completa.

- Aplicaciones en red

  TCL tiene un modelo de programación orientado a eventos que hace que la programación de clientes y servidores se simplifique. Eventos sobre ficheros, red e interfaz de usuario funcionan todos de la misma forma, haciendo que el estilo de programación sea consistente, potente y fácil de aprender, sin la necesidad de librerias adicionales.

- Gratuito

  TCL es código abierto, disponible de forma gratuita, lo que implica que se puede hacer lo que se quiera con el código, desde utilizarlo en aplicaciones comerciales sin licencias hasta modificar el propio código para que se adapte a las necesidades personales.

2.2. Dotlrn

2.2.1. Introducción

La plataforma de código abierto .LRN[1] ha sido desarrollada para albergar comunidades virtuales que aglutinan usuarios con intereses comunes y que comparten recursos, dispositivos, servicios etc. Este concepto de comunidad virtual puede utilizarse además para dar suporte a experiencias educativas tales como cursos online o de apoyo a cursos presenciales. Una de las ventajas de esta plataforma está en que es de código abierto (Open Source), lo que implica que el desarrollo sobre .LRN y su despliegue está al alcance de cualquiera. El ámbito de diseminación de esta herramienta consta de más de medio millón de usuarios en todo el mundo.

2.2.2. Teleeducación

La teleeducación o e-learning como se conoce por su término en inglés tiene cada vez más importancia debido a las ventajas que ofrece frente a la educación tradicional, especialmente cuando nos referimos al ámbito superior de enseñanza (grado y postgrado). Entre sus ventajas, se incluyen el ahorro de costes de personal docente y la accesibilidad por parte de la población a cursos especializados ya que los alumnos de un curso pueden estar dispersos geográficamente.

La educación tradicional sigue el modelo síncrono donde el profesor y los alumnos coinciden en tiempo y espacio. Sin embargo, la teleeducación permite modelos asíncronos en tiempo y espacio, utilizando sistemas de videoconferencia y trabajo colaborativo para sustituir la interacción directa entre profesor y alumnos que se produce en una metodología de aprendizaje síncrona.

Otra posibilidad de los sistemas de teleeducación es la capacidad para permitir el autoestudio. En este caso son los propios alumnos los que deciden la planificación del estudio, permitiendo una mayor flexibilidad que la que ofrece el sistema síncrono.

Esta metodología de estudio es muy interesante para las universidades no presenciales, pero también lo es para las presenciales ya que permite dotar de una mayor libertad a los alumnos a la hora de estudiar y también favorece la interacción entre los alumnos, lo que permite que tengan realimentación durante el proceso de aprendizaje.

Diversas plataformas de teleeducación han aparecido en los últimos años con el fin de ofrecer un servicio integral para desplegar asignaturas y cursos

[1] www.dotlrn.org

a distancia. Entre los más importantes podemos encontrar además de .LRN a:

- Moodle[2]

  Moodle es un acrónimo de Modular Object-Oriented Dynamic Learning Environment (Entorno de Aprendizaje Dinámico Orientado a Objetos y Modular). También es un verbo que describe el proceso de deambular perezosamente a través de algo o hacer las cosas según se te ocurre hacerlas. [5]

  - Paquete de software creado para la realización de cursos y sitios Web en internet.
  - Se distribuye gratuitamente como software libre.
  - Proyecto en desarrollo diseñado para dar soporte a un marco de pedagogía construccionista social.
  - Escrito en PHP y soporta varias bases de datos (la más utilizada MySQL)
  - No reutiliza código de otras aplicaciones como pudiese ser la gestión de usuarios o la gestión de contenidos

- Blackboard[3]

  Sistema propietario desarrollado por educadores para educadores y con un repositorio global de contenidos en el que cualquier usuario puede publicar o buscar recursos de aprendizaje

  - Permite la habilitación de redes sociales mediante un servicio web denominado Scholar.com[4] que permite localizar personas de otras instituciones con intereses comunes.
  - Permite crear un sistema de gestión de datos colaborativo con el que las instituciones podrían compartir datos anónimamente con el objetivo de contrastar estrategias y programas de teleeducación.
  - Se usa para gestionar cursos, crear contenidos y actividades, y fomentar la colaboración.

- Claroline[5]

[2] www.moodle.org
[3] www.blackboard.com
[4] www.scholar.com
[5] www.claroline.net

Plataforma de aprendizaje y trabajo virtual de código abierto y software libre que permite a los formadores construir cursos online y gestionar las actividades de aprendizaje y colaboración en la web. [6]

- Capaz de albergar un gran número de usuarios fácilmente.
- Es compatible con los entornos de Linux, Mac y Windows.
- Escrito en PHP y utiliza la base de datos MySQL.
- Integra estándares actuales como SCORM e IMS-LD.
- Ofrece una interfaz sencilla e intuitiva.

Existen otras plataformas que no se van a detallar pero que también se deben mencionar como son: Dokeos[6], Proyecto Sakai[7], eCollege[8] y ATutor[9].

2.2.3. Diseño y arquitectura de .LRN

El framework orientado a objetos de OpenACS es utilizado por .LRN para la creación de aplicaciones web. Esta plataforma también ofrece un repositorio (content repository) donde se almacenan los módulos de gestión de usuarios y grupos usado por el sistema de permisos, lo que permite tener un sistema con persistencia, capacidad de autenticación y el uso de plantillas. En los siguientes puntos se dará una visión más detallada tanto de la base de datos que utiliza la plataforma como de OpenACS, el framework que sustenta a .LRN.

2.3. OpenACS

Para la realización de este proyecto se realizó un estudio de Dotlrn, ya que es la plataforma de teleeducación sobre la que se quería trabajar y después se realizó la implementación de una aplicación lo más generalista posible, lo cuál hizo que para su desarrollo se utilizaran únicamente las características de OpenACS[10], dejando .LRN como herramienta para la utilización de dicha aplicación, aunque esta pueda ser utilizada sin tener .LRN instalado.

El origen de OpenACS se haya en la época de aparición de las empresas puntocom. En dicha época, cualquier empresa que trabajara sobre nuevas

[6] www.dokeos.com/es
[7] sakaiproject.org
[8] www.ecollege.com
[9] www.atutor.ca
[10] www.openacs.org

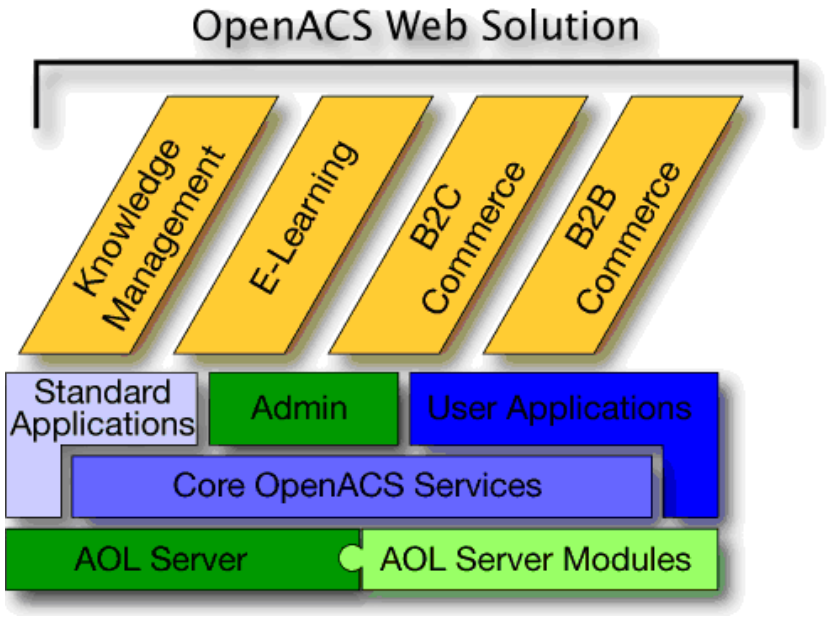

Figura 2.1: Esquema de OpenACS (Fuente: consulting.pumptheory.com)

tecnologías o internet podía encontrar financiación para sus proyectos de fácilmente. Es entonces cuando se comienza a hablar sobre comunidades virtuales y las oportunidades que estas ofrecen, sobretodo en el ámbito comercial. Sin embargo, en la actualidad ya no es tan sencillo crear un nuevo modelo de negocio basado en comunidades virtuales.

Para crear comunidades virtuales es necesario gestionar contenidos dinámicos, y para ello es necesario utilizar una base de datos relacional y un lenguaje de programación. Este trabajo no presenta una gran complejidad técnica, sin embargo, repetir el trabajo necesario para su creación con cada nuevo proyecto no es lógico, ya que el programador debe repetir dicho trabajo, generando una carga de trabajo adicional y que se refleja en el coste final que repercute directamente en el cliente.

En la mitad de la década de los 90, Philip Greenspun escribió unas herramientas modulares que ofrecían la capacidad de crear comunidades virtuales. Dichas herramientas fueron la base de su tesis doctoral y le permitieron crear la empresa Arsdigita. Dicho conjunto de herramientas se llamó Arsdigita Community System (ACS) y se distribuyeron con licencia GNU/GPL (General Public License). Como base de datos, Greenspun eligió Oracle debido a las prestaciones que esta le ofrecía.

ACS es una herramienta muy potente, sencilla y con la capacidad de resolver proyectos grandes, tales como la intranet del departamento de ventas

de Siemens (con 10000 empleados en 35 países) y el sistema Global Development Gateway del Banco Mundial. Pese al éxito de este paquete de herramientas, la empresa que fundara Philip Greenspun desapareció debido a problemas de gestión, quedándose Red Hat con los activos que quedaron tras el cierre de Arsdigita.

Aquí podría acabar la historia, sin embargo, al liberarse ACS bajo licencia GNU/GPL, la comunidad de desarrolladores retomó el proyecto donde se había parado al cerrar Arsdigita. Es en este punto, donde se decide utilizar una nueva base de datos que suplantara a Oracle por una de código libre (Postgresql) creando lo que ahora se conoce como OpenACS.

2.3.1. Ventajas de OpenACS

Una de las ventajas de OpenACS ya comentada en el capítulo de introducción de la memoria es que se trata de una plataforma de código abierto, pero existen otras ventajas por las cuales es la opción preferida y que se enumeran a continuación:

- Base de datos con alto rendimiento y escalabilidad. Algunos de los mayores websites del mundo funcionan con la tecnología de OpenACS tales como aol.com, netscape.com, moviefone.com, etc.

- Tecnología madura. OpenACS utiliza un fondo de conexiones a la base de datos que reduce el tiempo de inicio y finalización de tareas. Este es parecido al funcionamiento de JDBC, pero anterior a él. Así mismo, OpenACS corre sobre un servidor multihilo como Apache, pero también lo precede por varios años.

- Fácil y flexible uso del caché para mejorar el funcionamiento de las aplicaciones.

- Alta fiabilidad. OpenACS utiliza las bases de datos relacionales Oracle y PostgreSQL. Ambas bases de datos pasan el test ACID, el cuál es importante para asegurar la integridad de los datos. Los dos sistemas de bases de datos soportan transacciones, integridad en las referencias y permiten que se ejecuten desde fuera de la propia base de datos utilizando un lenguaje de programación.

- Componentes implementados en paquetes, lo cuales son de fácil instalación y actualización.

- Internacionalización, OpenACS permite la traducción del paquete al idioma del usuario de forma sencilla.

- Repositorio de contenidos y sistema de control de contenidos completamente funcional.
- Lenguaje de script ligero, sencillo y rápido con un API de fácil uso para generar websites desde la base de datos.
- Testeo automatizado (OpenACS permite crear baterías de pruebas de forma automatizada utilizando el lenguaje de script).
- OpenACS ofrece un gran número de aplicaciones ya creadas y de alta calidad como: E-commerce, Blogger, Chat, Forums, Project-manager, Calendar, Webmail, etc.
- Sistema de objetos que se encuentra sobre la base de datos, permitiendo a los desarrolladores a crear aplicaciones complejas usando objetos del API.
- Documentación extensa incluida en el paquete de herramientas.
- OpenACS es open source (código libre), lo que implica reducir gastos en desarrollo y para el cliente al no tener que pagar licencias.

2.3.2. Comunidad de OpenACS

Es de destacar la comunidad de desarrolladores de OpenACS, con más de 13000 miembros. Esta comunidad es especiamente activa ya que muchos de sus miembros se conectan diariamente y de gran utilidad ya que se pueden reducir tiempos de bloqueo durante la implementación de una aplicación gracias a la ayuda de otros miembros de la comunidad.

Una de las ventajas ya comentadas anteriormente es la cantidad de documentación que hay sobre OpenACS y esto se debe principalmente a las contribuciones de los miembros.

2.3.3. Desarrollo de aplicaciones en OpenACS

OpenACS es una plataforma a la que agregar una nueva aplicación es equivalente a escribir un nuevo paquete que contendrá la mayor parte de la lógica de la nueva aplicación pudiendo reutilizar partes de otros paquetes para el diseño del nuevo. Los paquetes también pueden ofrecer servicios sin llegar a ser aplicaciones completas en cuyo caso puede que no tengan páginas asociadas al mismo pero normalmente los paquetes son a la vez servicios y aplicaciones por lo que ofrecen una serie de funcionalidades que pueden ser utilizadas por otros paquetes y también ofrecen dichas funcionalidades

ellos mismos mediante un conjunto de páginas. Dichas páginas se sirven de un sistema de plantillas para facilitar su construcción y utilizan la tecnología ADP ofrecida por el servidor AOLserver para crear las páginas con contenido dinámico.

Muchas de las aplicaciones desarrolladas para OpenACS tienen como objetivo su uso en comunidades virtuales. Para ello se crean *portlets* que se incluyen dentro de la página de la comunidad (figura 2.2). Estas comunidades virtuales se crean con el objetivo de intercambiar información entre usuario con inquietudes similares o que comparten una afición o estudios, también ofrecen apoyo para el desarrollo de proyectos o cursos.

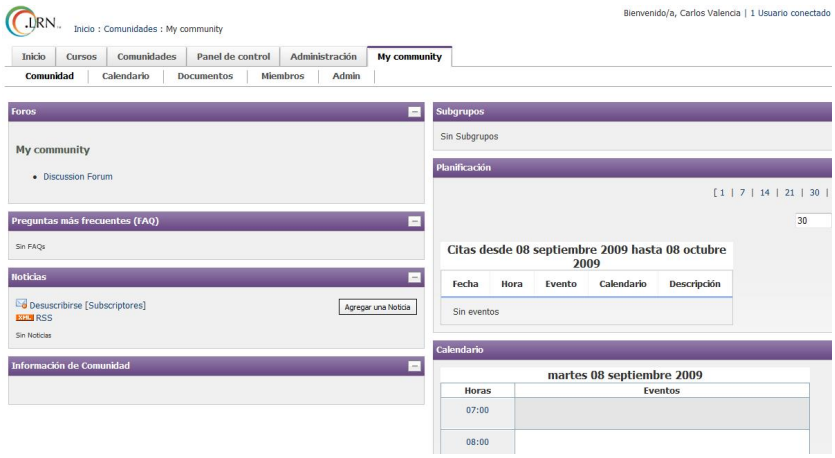

Figura 2.2: Captura de pantalla de comunidad virtual en OpenACS

OpenACS ofrece la posibilidad de utilizar objetos y para ello se utiliza el *Content Repository*. Este ofrece un soporte para versiones, organización jerárquica de permisos y extensión de tablas de forma dinámica. Por estas razones el uso del *Content Repository* es muy útil en la mayoría de las aplicaciones aunque hay casos en los que no merece la pena usarlo si no se necesitan guardar versiones ya que en ese caso se reduce la carga de la base de datos al no tener que guardar la información de todas las versiones ni información adicional a la propia de los datos necesarios en la aplicación.

Además de las ventajas anteriores, el *Content Repository* se utiliza en muchas ocasiones porque se pueden tratar los conjuntos de datos como objetos en lugar de como filas de tablas. De este modo la programación se asimila más a lenguajes de programación como Java por lo que facilita la implementación de aplicaciones a muchos desarrolladores. Este uso de objetos es muy útil porque se pueden agregar nuevos datos al mismo objeto sin necesidad de modificar el modelo relacional de la base de datos ya que de ello se encarga el sistema.

2.4. AOLserver

AOLserver es el servidor sobre el que se monta OpenACS. A priori, se podría preguntar por qué no usar Apache, el servidor más utilizado en la actualidad (usado por el 66 % de los sitios con más tráfico). Las razones para elegir Apache serían claras: flexibilidad, seguridad, estabilidad, portabilidad, es extensible, cumple con los estándares de internet, gratuito y código libre (open source).

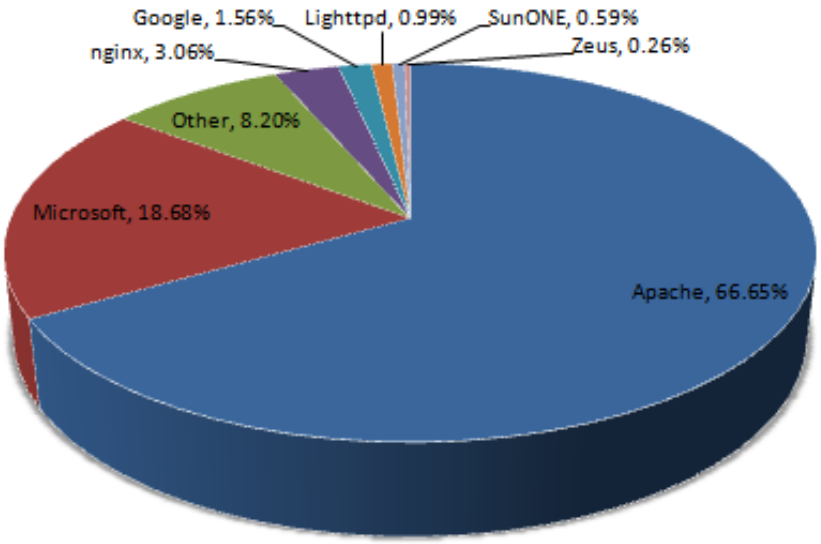

Figura 2.3: Porcentajes de utilización de servidores (Marzo 2009)

¿Por qué se ha utilizado AOLserver?

AOLserver tiene características similares a Apache en cuanto a que es código libre, fácil de configurar, la flexibilidad y que ambos tienen APIs para escribir módulos en C y TCL. Sin embargo, ambos servidores se diferencian en su estructura, haciendo que AOLserver sea más eficiente para determinados casos.

AOLserver tiene una arquitectura multihilos que incluye un intérprete de TCL, también tiene una API de la base de datos y un fondo (pool) de conexiones a la base de datos. Todo esto unido a un sistema para crear páginas dinámicas denominado ADP (AOLserver Dynamic Pages) hace que AOLserver sea elegido como servidor en sitios web que requieren numerosas conexiones a la base de datos y con personalización de páginas para cada usuario.

AOLserver fue desarrollado en primera instancia por NaviSoft y se llamó NaviServer. En 1995, American Online (AOL) compró la compañía

y cambió el nombre del servidor al que ahora se conoce. Posteriormente, en 1999, Philip Greenspun (diseñador y desarrollador de la primera versión de OpenACS llamada ACS) consiguió que AOL liberalizara el código del servidor.

Fue el primer servidor HTTP en combinar el procesamiento multihilo con un lenguaje intepretado y también el primero en utilizar procesamiento de colas de conexiones persistentes (Connection pools) con la base de datos.

2.4.1. AOLserver Dynamic Pages (ADP)

Una vez introducido el servidor y explicada brevemente su evolución histórica, ahora se va a tratar uno de los aspectos fundamentales por lo que se utiliza, se trata de las llamadas AOLserver Dynamic Pages (ADP) o páginas dinámicas.

Ahora existen otras posibilidades para crear páginas dinámicas, como son ASP (Active Server Pages) y JSP (Java Server Pages). Ambas se basan en páginas escritas en HTML en las que se incluyen fragmentos de un lenguaje de programación para obtener recursos personalizados para cada visualización de la página. Aún así, OpenACS sigue utilizando las páginas ADP puesto que está escrito en TCL mientras que para los otros casos se utiliza VBScript para ASP y Java para JSP.

2.5. Base de datos

OpenACS puede utilizar tanto Oracle como PostgreSQL[11] como base de datos relacional. Esto se debe al origen de OpenACS ya que su autor decidió utilizar Oracle debido a sus prestaciones. Posteriormente, la comunidad de desarrolladores de ACS decidieron crear una nueva versión que utilizase sólo código libre y por eso se decidió utilizar PostgreSQL y, aunque en la actualidad soporta ambas bases de datos, en realidad se suele utilizar PostgreSQL puesto que no se debe pagar ninguna licencia para su uso, lo que reduce el presupuesto de los proyectos implementados sobre OpenACS.

A continuación se exponen las comparaciones de PostgreSQL frente a otras bases de datos:

PostgreSQL frente a Interbase[12]

- PosgreSQL es gratuito mientras que Interbase es comercial
- PosgreSQL está mejor documentado

[11] www.postgresql.org

[12] www.codegear.com/products/interbase

- Soporta un número ilimitado de filas, índices y 1Gb por campo.
- Tiene un campo booleano
- Soporta herencia
- Permite indexar textos para realizar búsquedas sofisticadas a través de OpenFTS
- Soporta caídas del sistema
- Soporta funciones en varios lenguajes y índices sobre funciones
- Tiene secuencias
- Cumple los estándares mejor que Interbase
- Tiene soporte para conexiones encriptadas SSL

PostgreSQL frente a MySQL[13]

- Es más rápido y eficiente que MySQL
- Tiene vistas (views)
- Soporta tamaños de filas y bases de datos ilimitados
- Soporta herencia
- Tiene soporte para unicode
- Resiste caídas del sistema
- Tiene triggers y rules
- Soporta subqueries, índices sobre funciones
- Tiene soporte para conexiones encriptadas SSL

PostgreSQL frente a Oracle[14]

- Es gratuido y de código abierto
- Tiene un esquema de licencia más sencillo que Oracle
- Soporta herencia

[13] www.mysql.com
[14] www.oracle.com/database/index.html

- Permite indexar textos para realizar búsquedas sofisticadas a través de OpenFTS
- Puede ser traducido
- Tiene soporte para conexiones encriptadas SSL

2.6. Servicios de gestión de *bookmarks*

Para entender que es un servicio de gestión de *bookmarks* podría valer que sería el equivalente a los favoritos que tenemos en un navegador pero al que se puede acceder desde cualquier ordenador del mundo ya que la información no se encuentra almacenada en nuestro ordenador sino remotamente en los servidores del servicio. Además, también se puede acceder a los favoritos de otros usuarios, así como obtener listas de los más recientes, los más populares, etc.

La anterior podría ser una respuesta válida, sin embargo, no es del todo correcta. Los servicios de gestión de *bookmarks* aparecen en la web 2.0 como una funcionalidad englobada en los marcadores sociales. Los marcadores sociales son un método para clasificar y compartir información en internet, ya sean enlaces, imágenes, noticias o cualquier otra cosa que sea susceptible de ser etiquetada y compartida.

En un sistema de gestión de *bookmarks* los usuarios guardan una lista de enlaces de Internet que consideran útiles. Dichas listas pueden accederse de manera pública o privada. Al poderse acceder de forma pública, se pueden ofrecer servicios que mejoran la búsqueda de información por parte de los usuarios al ofrecer enlaces que otros usuarios han etiquetado con las mismas etiquetas que lo que estamos buscando, además, también se pueden ofrecer *bookmarks* más populares, *bookmarks* relacionados porque hay muchos usuarios que tienen siempre varios *bookmarks* aunque estos tienen distintas etiquetas, los más recientes, etc. Estos servicios también suelen ofrecer feeds para obtener la información almacenada de los usuarios de modo que se puede incrustar en otras páginas o puede ser recopilada por otras aplicaciones para ofrecer un valor añadido a la información (este es el caso de la aplicación desarrollada en este proyecto).

El sistema de etiquetación de *bookmarks* tiene la ventaja de que los enlaces son etiquetados por humanos, por lo que la etiquetación es mejor que la hecha por máquinas, además, los usuarios suelen marcar aquellos enlaces que les resultan útiles por lo que se incrementa el valor de las búsquedas en estos sistemas.

Por otra parte, al no haber un sistema preestablecido de categorías o etiquetas hace que pueda haber enlaces etiquetados incorrectamente. Tampoco existe una definición de como deben crearse las etiquetas de modo que puede estar la misma palabra en singular y plural, no hay una jerarquía de etiquetas y puede que estas sean personales para un determinado usuario de modo que para otros usuarios no tengan ningún valor dichas etiquetas.

2.6.1. Listado servicios online

Existen multitud de servicios de gestión de *bookmarks* que han aparecido en los últimos años. A continuación se presentan los más usados (todos ellos son en inglés):

- Del.icio.us[15]

  Servicio de gestión de *bookmarks* sociales en web. Ofrece un servicio similar al que se ofrecía en los navegadores para la gestión de *bookmarks* pudiendo agregarlos y etiquetarlos para obtener una clasificación por categorías. Adicionalmente, nos ofrece la posibilidad de crear *bookmarks* privados, que son aquellos que sólo puede verlos el usuario que los crea y se pueden obtener mediante feeds si se tiene la clave correspondiente al usuario.

  Es simple y sencillo de usar, lo que permite ser utilizado en dispositivos con capacidad limitada como móviles y por todo tipo de personas, independientemente de sus conocimientos técnicos.

Figura 2.4: Logotipo de delicious

- Blinklist[16]

  Este es un servicio rápido y de fácil uso. Permite, al igual que delicious, crear *bookmarks* privados y también tiene un sistema de feeds basado en RSS. Dispone de nubes de etiquetas con los tags generales y los personales. Tiene una ventaja respecto a delicious y es que permite importar los *bookmarks* almacenados en el navegador, así como otros almacenados en otros sistemas de gestión de *bookmarks* (delicious y

[15] www.delicious.com
[16] www.blinklist.com

Furl). Al ser un servicio social, también permite agregar contactos que pueden ser de dos tipos: amigos y fans. Los amigos son aquellos que tu has agregado y los fans son los que te agregan a ti.

Si se compara Blinklist con delicious se ve como Blinklist es un poco más fácil de usar y más potente que delicious, sin embargo, deliciuos es más ligero lo que significa que no consume tantos recursos y por tanto carga más rápido, además de tener un interfaz más sencillo lo que permite ser utilizado más fácilmente en dispositivos con capacidad limitada.

- Stumbleupon[17]

  Página comercial que integra una red social con la capacidad de intercambiar páginas de interés entre los usuarios de manera online mediante la utilización de una barra de herramientas que se incrusta en el navegador (actualmente disponible para Firefox, Mozilla Application Suite e Internet Explorer).

- Connotea[18]

  Similar a del.icio.us en cuanto a forma pero especializado en las áreas científica y clínica.

- CiteUlike[19]

  Servicio destinado a investigadores que tiene la capacidad de almacenar, organizar y compartir los artículos y las publicaciones de interés que leen.

- Diigo[20]

  Menos potente que los anteriores. Destaca por su interfaz, similar a la de un servidor de correo. También ofrece un servicio diferente a los demás ya que utiliza PeopleRank con lo que muestra los enlaces más populares basándose no sólo en el número de usuario que tienen dicho enlace sino que además también tiene en cuenta la fiabilidad de un determinado usuario.

  Ofrece una barra de herramientas para incorporar a nuestro navegador que nos permite, además de guardar páginas de interés, seleccionar

[17] www.stumbleupon.com
[18] www.connotea.org
[19] www.citeulike.org
[20] www.diigo.com

textos de páginas y almacenarlos para consultarlos posteriormente y tiene la opción de enviar texto para publicar en un blog.

Desde el punto de vista de la educación, ofrece cuentas premium para educadores con las que se pueden crear cuentas a los alumnos directamente sin que ellos tengan que hacerlo de manera independiente, las políticas de privacidad en las cuentas de los alumnos sirven para que estos sólo puedan comunicarse con otros usuarios y con los profesores; todos los alumnos de una clase se convierten en un grupo sin necesidad de crearlo y los anuncios que muestra la plataforma se reducen sólo al ámbito académico.

También existen servicios en español, aunque su uso es menor que los mencionados anteriormente:

- ifavoritos[21]

  Similar a del.icio.us. Permite guardar direcciones y acceder a ellas más tarde desde cualquier ordenador siempre que el navegador admita cookies. Dichos *bookmarks* tienen etiquetas asociadas con las que se ofrece la posibilidad de encontrar webs similares basandose en la etiquetación que los usuarios hacen en cada *bookmark*. Actualmente, este gestor de *bookmarks* se encuentra en versión Alfa por lo que se sigue mejorando el servicio.

- Mister Wong[22]

  Ofrece la posibilidad de tener marcadores privados y públicos, poder añadirse a grupos de interés, etc.

- favoriting[23]

  Comunidad libre de usuarios que comparten *bookmarks*.

2.6.2. Razones para la elección de delicious

La primera razón por la que se ha elegido del.icio.us, y quizá la más importante de todas, es que se trata del servicio de gestión de marcadores más conocido, y esto implica que es el que tiene más usuarios. Es posible que no sea el mejor desde el punto de vista de servicio, pero si se quiere realizar una aplicación que la gente vaya a utilizar, debemos dar funcionalidad a partir de aquello que usa la gente. Esta razón es suficiente para elegir del.icio.us, pero

[21] www.ifavoritos.com
[22] www.mister-wong.es
[23] www.favoriting.com

si también decimos que es un sistema sencillo de usar y además nos permite sindicar con un poderoso sistema de RSS, entonces la elección es mucho más sencilla. Además, al ser delicious el gestor de *bookmarks* más usado, hace que los demás tengan exportadores para introducir los *bookmarks* almacenados por un usuario en delicious por lo que simplifica la tarea a quellas personas que tengan sus *bookmarks* almacenados en un gestor diferente.

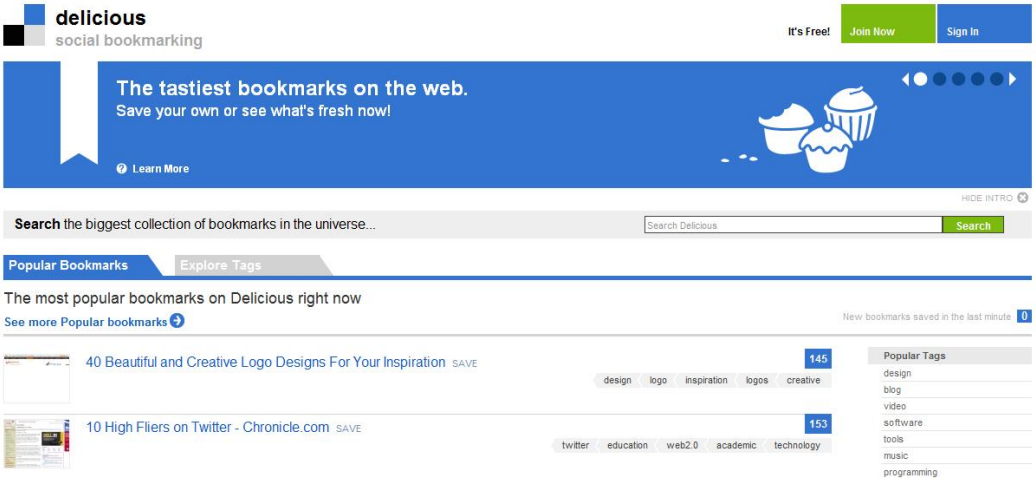

Figura 2.5: Captura de pantalla de la página principal de delicious

RSS en delicious

Antes de entrar en detalle sobre como están construidos los feeds obtenidos de delicious por RSS, primero hay que hacer una pequeña explicación para saber que es RSS.

RSS[24] son las siglas de Really Simple Syndication, que en castellano significa sindicación de contenidos de páginas web. RSS es un método para extraer contenidos de una página o un servicio web para utilizarlos en otros servicios o páginas web. Este es el vehículo elegido para extraer información relevante de *bookmarks* de las cuentas de usuario de los miembros registrados en la comunidad para su almacenamiento en la base de datos para su posterior uso.

Debido al gran volumen de información que trata el servicio de gestión de *bookmarks* de delicious, especifican que los feeds no se actualizan inmediatamente que se produce un cambio.

A continuación se explica la estrucutra de un feed como los que se obtienen de delicious:

[24] www.rssboard.org

Primero se encuentra el elemento raíz rss con versión 2.0. Este elemento es el que declara que el archivo XML recibido es un feed de rss.

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0" xmlns:atom="http://www.w3.org/2005/Atom"
 xmlns:content="http://purl.org/rss/1.0/modules/content/"
 xmlns:wfw="http://wellformedweb.org/CommentAPI/"
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns:cc="http://web.resource.org/cc/">
```

Contiene la etiqueta channel que está definida en la especificación de rss 2.0 y que contiene los items que vamos a obtener de delicious así como tres elementos obligatorios: title (nombre del canal, indica como se denomina el servicio ofrecido), link (URL del sitio web al que se refiere el feed) y description (descripción o frase explicativa del feed).

```
  <channel>
    <title>
      Delicious/pruebaPFC/telematica
    </title>
    <link>
      http://delicious.com/pruebaPFC/telematica
    </link>
    <description>
      bookmarks tagged telematica by pruebaPFC
    </description>
```

Después de estos elementos obligatorios tenemos etiquetas item que son las que contienen toda la información relacionada con los *bookmarks*. En cada etiqueta item se almacena la información de un *bookmark*. A continuación se explican los elementos que nos interesan que están contenidos en la cada item (a partir de aquí se utilizará la palabra episodio para referirse a cada item):

- title

  Nombre de la página web a la que se hace referencia, puede ser el nombre que aparece en la barra del navegador cuando se accede a dicha página o cualquier otro título que se escriba manualmente.

  ```
  <title>
      Departamento de Ingeniería Telemática - UC3M
  </title>
  ```

- pubDate

  Fecha en la que se añade el *bookmark* en delicious.

```
<pubDate>
  Fri, 21 Nov 2008 09:08:29 +0000
</pubDate>
```

- guid

  Identificador único que se asocia a cada episodio, con este elemento se puede comprobar si un determinado *bookmark* ha sido modificado ya que si el guid es el mismo significa que no se ha modificado nada pero si cambia, entonces significa que se han producido cambios.

```
<guid isPermaLink="false">
  http://delicious.com/url
   /188ed9c3c6eed1f0de8e68cff76fb38a#pruebapfc
</guid>
```

- link

  URL de la página web a la que está asociado el *bookmark*.

```
<link>
  http://www.it.uc3m.es/
</link>
```

- category

  Etiquetas que se asignan a cada *bookmark* para su clasificación. Un item puede contener cero, uno o más elementos category.

```
<category domain="http://delicious.com/pruebapfc/">
  uc3m
</category>
<category domain="http://delicious.com/pruebapfc/">
  universidad
</category>
<category domain="http://delicious.com/pruebapfc/">
  telematica
</category>
```

Construcción de URL de RSS en delicious

Delicious ofrece un amplio rango de posibilidades para la obtención de feeds procedentes de delicious. Lo primero que ofrece son dos sistemas para la obtención de dicha información (RSS y JSON), aunque en esta memoria sólo trataremos RSS ya que es el que se ha elegido.

Todas las URL que se construirán tendrán como prefijo `http://feeds.delicious.com/v2/rss`. Si se envía este prefijo sin añadir elementos en la ruta se recibe un feed con los *bookmarks* incluídos en el hotlist de delicious.

Antes de explicar las distintas posibilidades que nos ofrece delicious, hay que comenta un elemento importante, el atributo count. Este atributo nos permite decidir el número máximo de *bookmarks* que vamos a recibir en un feed y funciona de la siguiente manera: habría que concatenar al final de la URL construida ?count=número_episodios donde número_episodios se sustituye por un número de 1 a 100 que representa el número máximo de elementos que se desea recibir. Por defecto este valor es de 15. Con este atributo podemos controlar la cantidad de información que recibimos ya que en la aplicación se da la libertad al administrador de la comunidad para elegir el número de episodios que se recibirán por feed, de este modo se puede controlar la cantidad de información que se obtiene para su posterior manejo. Además, incluir más episodios en un feed implica que estos son más antiguos, por esta razón puede ser conveniente reducir su número para que la información que se presente a los usuarios sea lo más reciente posible y por tanto tenga mayor valor.

A continuación se enumeran los distintos sufijos que se pueden agregar al prefijo visto anteriormente y el resultado que obtendríamos:

- /recent

  Obtenemos los *bookmarks* más recientes que se han incluído en deliciuos por parte de alguno de sus usuarios.

- /popular

  Obtenemos los *bookmarks* más populares en delicious.

- /etiqueta

  Obtenemos *bookmarks* que estén etiquetados con la palabra que hayamos puesto en lugar de etiqueta.

- /nombre_usuario

  Obtenemos los últimos *bookmarks* que ha publicado el usuario seleccionado.

- /nombre_usuario?private=key

  Obtenemos los *bookmarks* privados del usuario seleccionado.

- /nombre_usuario/etiqueta[+etiqueta+...+etiqueta]

  Obtenemos los *bookmarks* del usuario seleccionado que esten etiquetados con las palabras por las que sustituyamos la palabra etiqueta.

Existen más opciones pero estas son las más interesantes para el proyecto que se desarrolla en esta memoria. La posibilidad de obtener *bookmarks* que contengan etiquetas determinadas es una opción muy potente ya que permite realizar un primer filtrado de los *bookmarks* que tiene cada usuario en su cuenta de delicious.

2.6.3. Consejos para un buen etiquetado

Este es un punto importante, ya que utilizar un buen sistema de etiquetado incrementa el valor de la información. Siguiendo las directivas que se exponen a continuación, los servicios de gestión de marcadores serán mucho más útiles. Y para la utilización de la aplicación desarrollada en este proyecto, un sistema de etiquetas coherente es necesario ya que se va a tratar la información de un grupo de usuarios de forma conjunta para obtener mejores resultados.

- Utilizar todas las etiquetas necesarias.
- Más globalidad: usar etiquetas en inglés.
- Orden de etiquetación: etiquetar desde lo más general hacia lo más particular.
- Las etiquetas son importantes, pero el título también lo es; y si este va acompañado de una buena descripción, el marcado ganará en utilidad para el resto de usuarios.
- Tener una etiqueta para cada concepto. Ejemplo: no tener para un periódico también la etiqueta diario.
- Etiquetas en singular. Ejemplo: no utilizar blog y blogs.

2.7. Mashups

Lo último que hay que comentar sobre el estado del arte se trata del tipo de aplicación que se ha desarrollado como parte de este proyecto. Dicha aplicación se puede incluir como *mashup*. Los *mashups* o aplicaciones web híbridas son aquellas que utilizan servicios externos (normalmente procedentes de otras aplicaciones web) para ofrecer nuevos servicios, consumiendo información procedente de dichos servicios externos.

Estos servicios se están haciendo cada vez más y más frecuentes al ofrecer servicios más especializados basados en servicios ya existentes. Dichos servicios ofrecen, a traves de interfaces públicas o usando APIs, la información necesaria para poder ofrecer los nuevos servicios. La facilidad a la hora de crear *mashups* ha hecho que cobren gran importancia en la web, y produce un efecto doble, ya que por un lado se ofrecen servicios muy útiles aunque también hay otros muchos que no pasan de ser novedosos pero sin éxito.

Capítulo 3

Descripción de la aplicación

Después de conocer el estado del arte y antes de comenzar con la explicación a fondo de los aspectos técnicos de los paquetes desarrollados, se debe describir la aplicación diseñada e implementada, que luego será explicada en dos partes atendiendo a los paquetes de software generados para implementar su funcionalidad.

3.1. Finalidad de la aplicación

La finalidad de la aplicación desarrollada consiste en el filtrado y posterior gestión de *bookmarks* utilizada en comunidades vituales de la plataforma de teleeducación .LRN. Obtiene información sobre *bookmarks* de cada uno de los miembros de la comunidad obtenidos de un servicio externo llamado del.icio.us. Filtra esa información en función de los requerimientos de la comunidad, expresados en forma de etiquetas que el administrador puede configurar, y almacena toda la información que cumple con dichos requerimientos. Dicho filtrado se realiza periódicamente para actualizar los datos almacenados y añadir nueva información.

Con toda la información de la que dispone la aplicación para una determinada comunidad, se muestra a los miembros de dicha comunidad y se ofrecen distintas opciones de listado en función de las preferencias del usuario atendiendo a la relevancia, antigüedad, etc.

Con carácter complementario y para dar mayor valor a la aplicación, ésta tiene un sistema de valoraciones que incrementa el valor de utilización de la misma, ya que utiliza la inteligencia colectiva doblemente. Por una parte obtiene los *bookmarks* de los miembros que tienen almacenados en del.icio.us, pero además, el sistema de votación permite clasificar aquellos *bookmarks* que han sido mostrados al resto de miembros de la comunidad por lo que se realiza

un filtrado adicional en el que se descubren cuáles son aquellos más útiles dentro del marco en el que se encuentre cada comunidad de usuarios.

3.2. Estructura de la aplicación

La aplicación se divide en dos paquetes de software que deben ser instalados para su correcto funcionamiento. El primero de los dos paquetes que se debe instalar es el paquete Rating. Este paquete ofrece un servicio de evaluación de objetos generalista; esto significa que se ha desarrollado a partir del diseño de la aplicación creada durante la duración de este proyecto, pero se ha aprovechado la generalidad de esta funcionalidad para crear un paquete que ofrezca soluciones a futuras aplicaciones y proyectos. El otro paquete es el paquete Bookmark, núcleo de la aplicación.

El paquete Rating ofrece una API para incluir sistemas de votación en aplicaciones basadas en la organización de OpenACS con la que se crean objetos que tienen un identificador único. Se ofrecen unos determinados sistemas que se explicarán más adelante. Sin embargo, se pueden generar otros sistemas de forma rápida y eficaz ya que el paquete tiene todo lo necesario para simplificar esta tarea. Por otra parte, este paquete también ofrece dos funcionalidades adicionales que suelen ir unidas a la evaluación de conceptos u objetos. Dichas funcionalidades son un sistema para albergar comentarios de los usuarios y otra de descripciones. Ambas dan mayor valor al paquete, al ofrecer todo aquello necesario para que los sistemas de votación se vean complementados con la capacidad de discusión entre los usuarios y la capacidad de exponer ideas. Se puede pensar que ambas funcionalidades son en realidad una única funcionalidad, pero estas no lo son ni a nivel técnico ni a nivel conceptual.

La página *descriptions* ofrece un entorno en el que exponer y desarrollar las ideas que tienen los usuarios. Cada usuario tiene la capacidad de publicar y editar un bloque de texto donde dar respuesta al tema expuesto, ofrecer su punto de vista, etc. Dichos bloques pueden ser valorados para ordenarlos de manera que los usuarios puedan leer primero aquellos que están mejor valorados por sus compañeros de la comunidad. La página *comments* ofrece un entorno donde publicar cualquier tipo de idea, pregunta u opinión que tenga un usuario. Los comentarios son mostrados uno detrás de otro para poder seguir el hilo de una discusión y los usuarios no tienen permiso para editar ni borrar dichos comentarios.

El núcleo de la aplicación se encuentra en el paquete Bookmark, este contiene todos aquellos procedimientos y páginas con las que se da servicio a la aplicación. Son de especial relevancia las páginas de perfil de usuario

porque son la cara pública de la aplicación. Estas se han creado de forma que queden lo más sencillas posible ya que de esta manera no se distrae el usuario con elementos superfluos que únicamente ornamentan la página ni información excesiva que reduzca la utilidad de la misma. Esta característica de sencillez de la aplicación se puede ver en la propia página de del.icio.us, pero también en muchas otras páginas de las más usadas de internet como Google.

Existen procedimientos ejecutados en *background* (segundo plano) que permiten la actualización de contenidos sin la necesidad de la existencia de un humano que realice dichas tareas. También existe otro procedimiento del mismo tipo para el control de votaciones de modo que no haya votaciones fraudulentas.

3.3. Perfiles de usuario

Existen dos perfiles de usuario en la aplicación, uno sin privilegios y otro con privilegios de administración.

Todos los procedimientos y páginas que tiene la aplicación se pueden clasificar en función del tipo de usuario que puede acceder a ellas. El primero de dichos grupos es la visión de usuario, en él se incluyen todos aquellos procedimientos que responden directamente a la interación de los usuarios con la aplicación y todas aquellas páginas a las que pueden acceder todos los usuarios. El segundo grupo sería aquel con perfil de administración. El perfil de administración de la aplicación ofrece una serie de páginas que no son mostradas a los usuarios sin privilegios y con las que se pueden modificar los parámetros de control de la misma. Este perfil es muy importante, ya que para sacar todo el rendimiento a esta aplicación se deben dar correctamente las etiquetas de filtrado y también es importante la monitorización de los *bookmarks* ya que los usuarios pueden añadir los *bookmarks* manualmente, por lo que en comunidades menos cerradas o menos profesionales puede haber miembros problemáticos que almacenen *bookmarks* fuera de contexto o con otra finalidad.

En la figura 3.1 se ve la página principal de la aplicación con un único *bookmark*. Después de las cabeceras que introduce OpenACS/.LRN se puede ver un enlace que dice: Ir a página de administración. Dicho enlace sólo es visible cuando el usuario tiene privilegios de administración y lo lleva a la página de administración del paquete Bookmark. En esta página también aparecen otra serie de botones y enlaces, los cuales se incluyen dentro del funcionamiento común a todos los usuarios. Existen dos botones que nos redirigen a la página donde se crean nuevos *bookmarks* manualmente y la

página de configuración de cada usuario y en cada fila de la tabla mostrada se ofrecen enlaces a la información detallada del *bookmark*, a la página a la que hace referencia, a la página para dejar comentarios y dos botones para puntuar dicho *bookmark* positivamente o negativamente.

Figura 3.1: Captura de pantalla de página principal de la aplicación

Posteriormente se explicarán todas las páginas de la aplicación con mayor detalle.

3.4. Internacionalización y localización

Una de las ventajas de utilizar OpenACS es la funcionalidad de internacionalización y localización que ofrece a través del paquete acs-lang. Este paquete provee un servicio de internacionalización del paquete que consiste en sustituir aquellos elementos de texto que se vayan a mostrar en las páginas de los paquetes desarrollados por claves que maneja el paquete acs-lang cuando un cliente pide una página para ofrecerlas en el idioma en el que esté configurada la plataforma, esto último se llama localización.

Como se puede concluir de la explicación anterior, además de ofrecer un sistema que muestre mensajes en distintos idiomas, también tiene la capacidad de dividir el trabajo en varias fases.

Una primera fase corresponde al desarrollador de la aplicación, el cuál debe seguir una serie de normas para que el paquete a desarrollar se internacionalice. En lugar en poner los mensajes que se muestran por pantalla dentro del código escrito en los ficheros .tcl y .adp, tendrá que sustituir estos por claves que son las que relacionan cada mensaje con la posición dentro de las páginas que se muestran al usuario.

Si el mensaje se encuentra en un fichero .adp se escribe en la posición en la que iría el mensaje #package.key# donde package es la clave asociada al paquete y key es la clave asociada al mensaje. Ejemplo:

```
<h1>#bookmark.Add#</h1>
```

Si el mensaje se encuentra en un fichero .tcl se escribe [_ package.key] en lugar del mensaje, teniendo package y key el mismo significado que en el los ficheros .adp. Ejemplo:

```
{title:text {label [_bookmark.Title]}}
```

La siguiente fase, aunque puede hacerse también en una cuarta fase para ampliar con nuevos idiomas la internacionalización del paquete, consiste en crear los archivos que contienen los mensajes en los diferentes idiomas. Se creará un archivo por cada idioma que se llamará con el nombre del paquete + . + idioma + .ISO-8859-1.xml (ejemplo: el nombre del fichero que contiene los mensajes en español del paquete *bookmarks* es bookmark.es_ES.ISO-8859-1.xml).

Los archivos de internacionalización comienzan siempre con un elemento raíz que tiene 3 atributos. El primero de los atributos, *package_key*, indica la clave de paquete, el atributo *locale* indica el idioma de los mensajes que contiene el fichero y *charset* contiene la codificación del archivo.

Cada mensaje es un campo de texto contenido dentro de una etiqueta msg con un atributo *key* que indica la clave de mensaje del texto que contiene.

A continuación se muestra un ejemplo de fichero de intenacionalización:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<message_catalog package_key="bookmark" locale="es_ES"
  charset="ISO-8859-1">
  <msg key="Add">Añadir</msg>
  <msg key="Yes">Sí</msg>
</message_catalog>
```

La tercera fase consiste en la localización y su ejecución corresponde al administrador de la plataforma. El administrador es el encargado de configurar la plataforma para que se muestre la aplicación en un determindado idioma.

3.5. *Breadcrumbs*

Como parte del diseño de la aplicación se ha incluído el uso de *breadcrumbs*. Los *breadcrumbs* son barras de información situacional dentro de una página web, se utilizan para dar información al usuario de donde se encuentra así como facilitar la navegación por la aplicación pudiendo ir directamente a los puntos que se incluyen en el *breadcrumb* en cada momento. En la figura 3.2 se muestra la posición del breadcrumb en la parte superior de la página a a derecha del logotipo de de .LRN.

Figura 3.2: Ejemplo del uso de breadcrumbs

3.6. Estructura de los paquetes desarrollados

Los paquetes desarrollados para esta aplicación siguen la estructura común de los paquetes desarrollados en OpenACS. Esta estructura permite almacenar el código generado de forma ordenada, facilitando el acercamiento a nuevo personal al código escrito por otra persona.

La estructura de los paquetes es la siguiente:

3.6.1. Archivo package.info

El fichero package.info para un determinado paquete se denomina cambiando package por el nombre del paquete conrrespondiente. en este fichero se almacena toda información relacionada con el paquete tal como su nombre, versión, nombre y dirección de correo electrónico del creador, una breve descripción de la utilidad del paquete, nombre del servicio que provee, nombre de los servicios que requiere para funcionar, etc. Pero también contiene

información a nivel de planificación de tareas con la definición de callbacks que son llamadas a procedimientos que se deben ejecutar en determinados momentos tales como en la instalación, desinstalación, etc.

Por último, también contiene la descripción de los parámetros de configuración que tiene el paquete para dar una funcionalidad más personalizada. Estos parámetros son los que permiten modificar tamaños máximos permitidos, especificar número de elementos, especificar funcionamiento de filtros y muchas más posibilidades en función de las necesidades que se descubran durante el proceso de diseño de la aplicación. Un punto importante respecto a los parámetros es que pueden ser de paquete o de instancia. Esto significa que en algunos casos estos parámetros necesita que tengan el mismo valor para cada una de las instancias del paquete en la plataforma y en otras ocasiones, es más útil que cada instancia tenga sus propios valores para dichos parámetros. Esto se puede ver de manera clara con parámetros que tengan que ver con etiquetas de filtrado para *bookmarks*, en este caso cada instancia del paquete pertenecerá a una comunidad distinta con necesidades distintas por lo que cada una de ellas tendrá unas restricciones distintas; por otra parte, un ejemplo de parámetro de paquete es el que limita el tamaño de un texto, ya que nos interesa que todas las instancias tengan la misma restricción para limitar el tamaño de almacenamiento necesario.

3.6.2. Carpeta catalog

Esta carpeta contiene los archivos generados para la internacionalización de los paquetes en OpenACS. Existirá un archivo por cada idioma que se denominará con el nombre del paquete + punto + el idioma (ejemplo: español de España es es_ES, inglés de EE.UU. es en_US e inglés de Gran Bretaña es en_UK) + .ISO-8859-1.xml. Con esta estructura, añadir un nuevo idioma consiste en duplicar uno de los ficheros existentes cambiando el identificador de idioma por el nuevo que queremos incluir y traducir todos los mensajes que vienen dentro del fichero xml al nuevo idioma.

3.6.3. Carpeta sql

Esta carpeta contiene a su vez dos carpetas: postgresql y oracle. Esta división en carpetas se debe al origen de OpenACS en el cuál se utilizaba Oracle como base de datos, pero al sacar una versión de código abierto, se decidió cambiar de base de datos para utilizar una de código abierto (PostgreSQL). Entre ambas bases de datos hay algunas diferencias al realizar determinadas tareas por lo que a veces es necesario tener archivos duplicados

que tengan en cuenta dichas diferencias para que los paquetes desarrollados funcionen con ambas bases de datos.

Dichas carpetas contendrán dos scripts: package-create.sql y package-drop.sql donde package se sustituirá por el identificador del paquete. El script package-create.sql contendrá las llamadas a la base de datos para crear todas las tablas necesarias para el funcionamiento del paquete, así como la creación de los tipos de objetos nuevos que se utilizarán en el paquete; y el script package-drop.sql contendrá las llamadas para eliminar todo lo que esté relacionado con el paquete en la base de datos.

Los scripts contenidos en esta carpeta son requeridos por el instalador de OpenACS durante el proceso de instalación del paquete.

3.6.4. Carpeta tcl

Esta carpeta contiene archivos tcl que contienen procedimientos que son usados por el paquete, y que pueden ser reutilizados por otros paquetes. Estos archivos contienen la parte del código que consume la mayor parte de la carga computacional del paquete.

3.6.5. Carpeta lib

Esta carpeta contiene páginas del paquete que puedes ser reutilizadas por otros paquetes.

3.6.6. Carpeta www

Esta carpeta contiene la mayor parte de las páginas que ofrece el paquete. La más importante de ellas es la página index.tcl e index.adp que es la primera página que se obtiene al entrar en la instancia del paquete.

Esta carpeta también incluye otras carpetas:

Carpeta Admin

Esta carpeta contiene las páginas de administración del paquete. La función del paquete es simplemente organizativa, no da ningún permiso o propiedad especial a las páginas que estén en esta carpeta. Para que las páginas almacenadas en esta carpeta sólo puedan ser accedidas por administradores debe hacerse la comprobación dentro del código de la página.

Carpeta resources

Esta carpeta contiene elementos auxiliares de presentación de las páginas del paquete tales como imágenes, iconos y diagramas. Los elementos contenidos en esta carpeta pueden ser reutilizados por otros paquetes.

3.7. Estrategia de desarrollo

Para afrontar un proyecto de cierta envergadura no se puede seguir la clásica técnica de codificar y corregir (code&fix), aunque es útil para pequeños proyectos por la sencillez para seguir dicha estrategia. Sin embargo, para proyectos más grandes, este ahorro en las fases de análisis, planificación, gestión de recursos y documentación es mucho menor que las dificultades que aparecen por ser el proyecto de mayor envergadura. Por esta razón se decidió seguir una metodología durante el desarrollo del proyecto.

La primera fase comenzó con una idea inicial que consistía en gestionar *bookmarks* obtenidos de un servicio externo en una comunidad de dotlrn. A partir de esta idea se estudiaron las necesidades que cubriría dicha aplicación y se formalizaron los requerimentos de la aplicación en las especificaciones del proyecto.

En la segunda fase, una vez definidas las especificaciones del proyecto, se cambió la estrategia para realizar el proyecto siguiento un ciclo de vida incremental, es decir, se implementó un primer prototipo con funcionalidad reducida y en sucesivos pasos se fue incrementando la funcionalidad del prototipo anterior. Esta metodología de trabajo ha permitido ir finalizando tareas de codificación de las distintas funcionalidades de la aplicación de forma independiente y obteniendo resultados visibles del trabajo a lo largo del periodo de desarrollo del proyecto.

Es destacable el uso de SVN como controlador de versiones ya que en todo momento se podía volver a una versión anterior así como almacenar los cambios organizándolos mediante comentarios en los que se explicaban dichos cambios, haciendo más sencillo el desarrollo conjunto de la aplicación en lugar de centrarse en un único punto de desarrollo.

3.8. Especificaciones

Como ya se ha mencionado en 3.7, en la fase inicial de diseño de la aplicación se definieron un conjunto de especificaciones que se incrementaron con los diferentes ciclos del proceso de desarrollo. A continuación se enumeran las especificaciones de la aplicación:

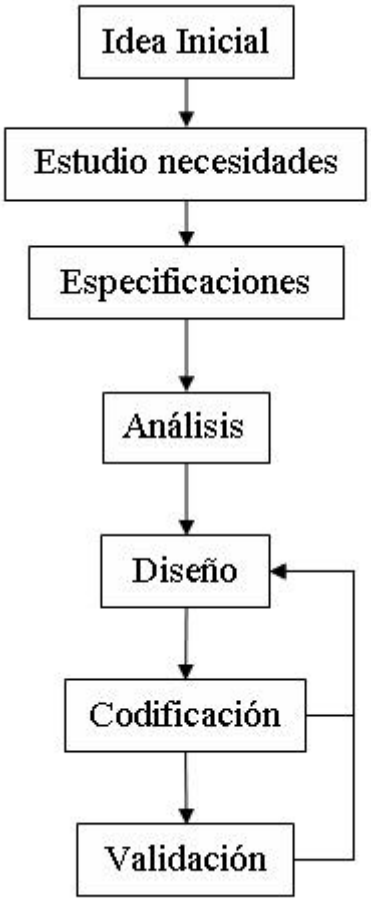

Figura 3.3: Ciclo de desarrollo

- Adquisición automática de *bookmarks* de las cuentas de usuario de delicious.
- Filtrado personalizado de *bookmarks* por instancia de la aplicación.
- Filtrado personalizado por usuario.
- Clasificación de *bookmarks* personalizado por usuario atendiendo a criterios de antigüedad y calidad (basado en un sistema de votaciones).
- Agregación manual de *bookmarks* para todos los usuarios.
- Sistema de votaciones para los *bookmarks* con control de votaciones múltiples.

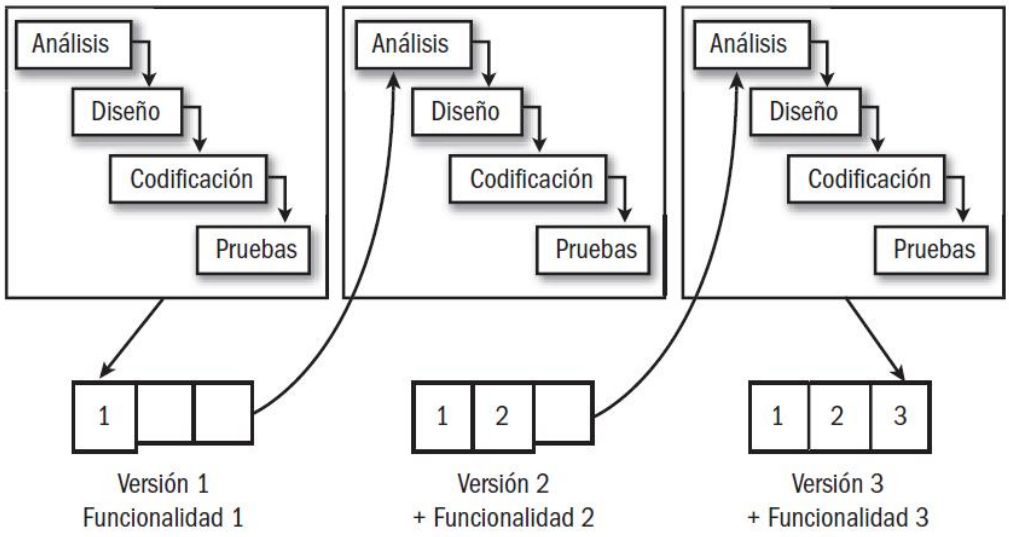

Figura 3.4: Ciclo de vida incremental

- Posibilidad de eliminación de *bookmarks* por parte de los administradores.
- Soporte para comentarios para cada *bookmark*.
- Soporte para descripciones para cada *bookmark*.
- Extensión del sistema de votación para la evaluación de descripciones de modo que se presenten las más relevantes primero.
- Extensión del sistema de votaciones con diferentes modelos de votación para dar carácter más generalista al paquete Rating.

Capítulo 4

Paquete Bookmark

Tras una fase de estudio de la plataforma dotlrn y del lenguaje de programación TCL, se procedió al análisis de los requerimientos de la aplicación que se iba a implementar como parte de este proyecto. Para ello se definió el paquete Bookmark.

4.1. Funcionalidad ofrecida por el paquete Bookmark

Este paquete es el núcleo del servicio de gestión de *bookmarks* para comunidades en OpenACS/.LRN. Por esta razón, la mayoría de la funcionalidad ofrecida en la aplicación desarrollada en este proyecto se encuentra dentro de este paquete, aunque también se ofrece funcionalidad adicional con la integración del paquete Rating, también desarrollado en este proyecto y que se explica en el siguiente capítulo.

Bookmark ofrece todos los recursos necesarios para la administración de un sistema de gestión de *bookmarks*. Para ello provee una página de administración para definir los parámetros necesarios para un funcionamiento más eficiente de la aplicación. También ofrece un servicio para prohibir determinados *bookmarks* que a juicio del administrador de la comunidad no sean aptos dentro de la comunidad, ya sea por tener contenidos inapropiados o de poca utilidad.

Bookmark también ofrece una página de configuración de usuario donde se permite a los usuarios personalizar los contenidos que se les muestran, ya sea eligiendo entre las distintas posibilidades que hay para mostrar los *bookmarks* por mayor votación, más novedosos o realizando filtrados adicionales.

También se ofrece una página para añadir *bookmarks* de forma manual. Esta página se ha creado ya que puede que haya usuarios que no tengan

cuenta en delicious pero que quieran participar en la comunidad. Podría haberse obligado a los usuarios a tener una cuenta en delicious pero es mejor ofrecer más alternativas a los usuarios ya que en ocasiones puede que no quieran dar información personal o sean reacios a usar otras aplicaciones. Además, un usuario que ya utiliza otro sistema de gestión de *bookmarks* puede que no quiera utilizar delicious y por tanto, tras contemplar todas las posibilidades anteriores, se concluyó que era mejor ofrecer esta página.

La página principal del gestor de *bookmarks* contiene un listado de los *bookmarks* más relevantes atendiendo a las preferencias de cada usuario y también enlaces, tanto a página de la propia aplicación como a las páginas a las que hacen referencia los *bookmarks*. Se ofrecen enlaces a páginas de información de todos los *bookmarks* en las que se detalla toda la información relativa al *bookmark* seleccionado. También se ofrece la posibilidad de hacer comentarios respecto a un determindado *bookmark* y desde la página de información de un determinado *bookmark* también se puede acceder a una página donde se puede realizar una descripción de la página a la que se hace referencia.

4.2. Modelo relacional de la base de datos

Como núcleo de la aplicación desarrollada, el modelo relacional de la base de datos del paquete Bookmark contiene toda la información imprescindible que ofrece esta aplicación. Por esta razón, se diseñó este modelo incluyendo todos aquellos casos de uso que se dan al paquete Bookmark, más la posibilidad de extender dichos casos sin la necesidad de modificar la estructura de la base de datos en el paquete.

Algunos campos de las tablas viene marcados con un asterisco (*), esto significa que dicho campo no se encuentra en dicha tabla pero su información forma parte de dicha tabla y no se ha incluído en ella por estar almacenado en la tabla acs_objects donde se crea una nueva entrada cada vez que se crea un nuevo objeto ya que esta tabla forma parte de la arquitectura de la plataforma OpenACS. Contiene, entre otros campos, el identificador único que OpenACS asigna a cada objeto creado, pero también información sobre el paquete en el que se creó el objeto, contexto, identificador del usuario que creó el objeto, fecha de creación, nombre del objeto y otros campos que también son de utilidad para la plataforma.

En la figura 4.1 se muestra un diagrama con el modelo relacional del paquete *Bookmark*. Después se detalla el contenido de cada una de las tablas explicando su uso dentro del paquete.

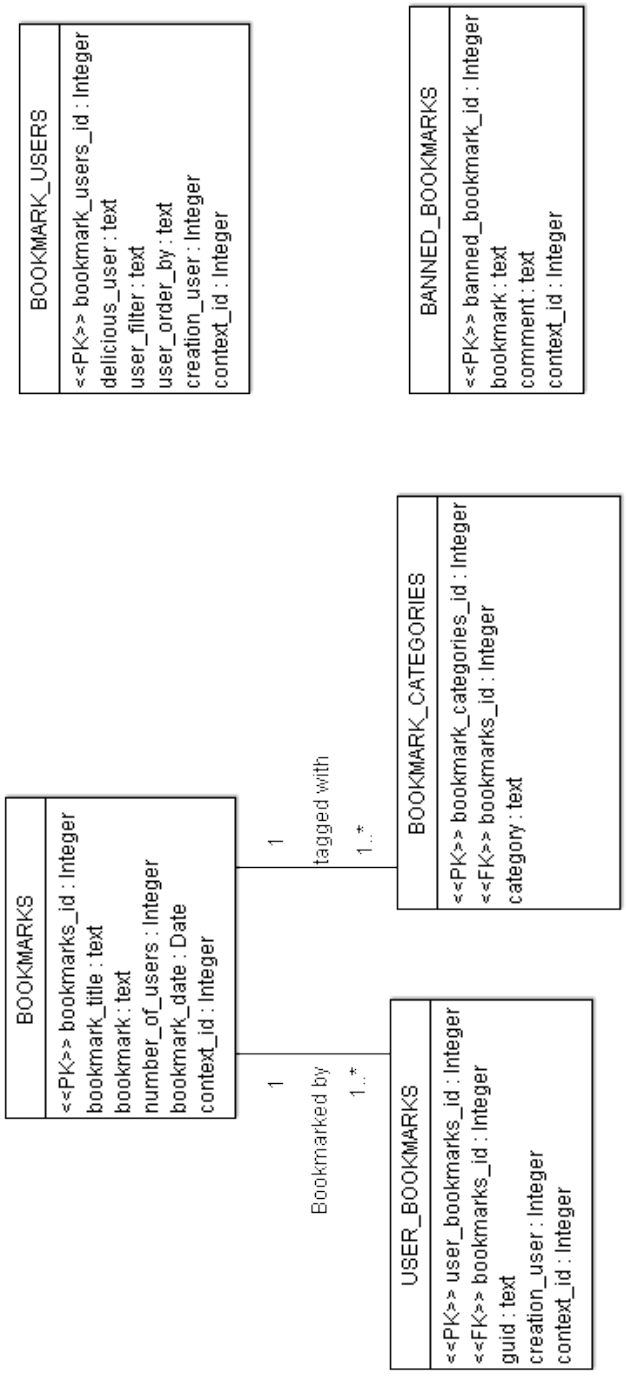

Figura 4.1: Modelo relacional del paquete Bookmark

4.2.1. Tabla bookmarks

Esta tabla contiene la información principal objeto de este proyecto, aunque no es suficiente para el correcto funcionamiento de la aplicación desarrollada. Puesto que se quiere dar un uso práctico y extensible a la aplicación de gestión de *bookmarks* se han incluido todos aquellos campos que pueden ser utilizados al almacenar un *bookmark*, aunque no siempre contengan información todos los campos.

Las columnas de la tabla bookmarks son las siguientes:

- bookmarks_id: integer, not null

  Este campo almacena el identificador que asigna OpenACS a cada nuevo objeto que se crea, este es único y se utiliza para mantener la integridad de la base de datos.

- bookmark_title: text

  Este campo contiene el título del *bookmark* que tenía guardado el usuario del que se obtuvo el *bookmark*. Si varios usuarios tienen el mismo *bookmark*, el título se corresponde al del primero del que se obtuvo la información del *bookmark*.

- bookmark: text

  Contiene el enlace al que hace referencia el *bookmark*. El enlace es formateado por la aplicación antes de almacenarlo para que no se produzcan duplicidades en los *bookmarks*.

- number_of_users: integer

  Número de usuarios que hacen referencia al *bookmark*. Esta información nos permite realizar una clasificación de los *bookmarks* en función del número de gente que los tiene almacenados. Cuanto mayor sea este número, más útil será dicho *bookmark* ya que por esta razón los usuarios lo tienen almacenado.

- bookmark_date: date

  Almacena la fecha de publicación del *bookmark*. Este dato nos sirve para poder realizar una clasificación de *bookmarks* en función de la antigüedad, suele ser útil dar primero aquellos *bookmarks* más nuevos ya que suelen ser lo que los usuarios todavía no han descubierto.

- context_id *

Este campo contiene el identificador de la instancia del paquete en el cuál se creó el objeto. Este dato nos permite diferenciar *bookmarks* de distintas comunidades para que puedan coexistir varias comunidades en la misma instancia de OpenACS pero que esto no interfiera en el correcto funcionamiento de cada una de ellas.

La clave primaria de la tabla es bookmarks_id. La tabla también tiene una clave externa, bookmarks_id con el campo object_id de la tabla acs_objects.

4.2.2. Tabla bookmark_categories

Esta tabla contiene las palabras clave asociadas a los *bookmarks*, esta información nos permite obtener filtrados personalizados de la información almacenada. Además, también nos permite tener una fuente de información respecto a que tipo de información está almacenada en función de las veces que se repitan cada una de las palabras clave.

Las columnas de la tabla bookmark_categories son las siguientes:

- bookmark_categories_id: integer, not null

  Este campo almacena el identificador que asigna OpenACS a cada nuevo objeto que se crea, este es único y se utiliza para mantener la integridad de la base de datos.

- bookmarks_id: integer

  Este campo sirve para relacionar cada fila de la tabla con el *bookmark* asociado. Este elemento contiene el identificador único de un objeto de la tabla bookmarks.

- category: text

  Contiene una categoría o palabra clave de un determinado *bookmark*. Existe una entrada en esta tabla por cada categoría que tenga un determinado *bookmark*.

La clave primaria de la tabla es bookmark_categories_id. La tabla también tiene dos claves externas, una es bookmark_categories_id con el campo object_id de la tabla acs_objects y la otra es bookmarks_id con bookmarks_id de la tabla bookmarks, esta clave externa nos relaciona los *bookmarks* con las categorías asociadas a ellos.

4.2.3. Tabla user_bookmarks

La tabla user_bookmarks almacena los usuarios que tienen marcados cada *bookmark*, así como el guid procedente de delicious. De esta forma se lleva el control de las actualizaciones de los *bookmarks* por parte de los usuarios en delicious, así como un control de los *bookmarks* añadidos manualmente dentro del entorno de la aplicación de gestión de *bookmarks*.

Las columnas de la tabla user_bookmarks son las siguientes:

- user_bookmarks_id: integer, not null

  Este campo almacena el identificador que asigna OpenACS a cada nuevo objeto que se crea, este es único y se utiliza para mantener la integridad de la base de datos.

- bookmarks_id: integer

  Este campo sirve para relacionar cada fila de la tabla con el *bookmark* asociado. Este elemento contiene el identificador único de un objeto de la tabla bookmarks.

- guid: text

  Contiene el identificador único asociado al episodio de rss obtenido de delicious para un determinado *bookmark* y un determinado usuario. En caso de que el usuario no tenga delicious o que haya agregado el *bookmark* a mano, este campo contendrá un símbolo de control para identificar dicho caso.

- creation_user *

  Este campo contiene el identificador del usuario que tiene almacenado este *bookmark* en su cuenta de delicious o el que lo agregó manualmente a la aplicación. Es necesario este campo para comparar el guid que recibimos en cada episodio con el correspondiente al usuario al que pertenece dicho episodio

- context_id *

  Este campo contiene el identificador de la instancia del paquete en el cuál se creó el objeto. Este dato nos permite diferenciar usuarios que pertenecen a distintas comunidades que usan el servicio de gestión de *bookmarks*.

La clave primaria de la tabla es user_bookmarks_id. La tabla también tiene dos claves externas, una es user_bookmarks_id con el campo object_id de la tabla acs_objects y la otra es bookmarks_id con bookmarks_id de la tabla

bookmarks, esta clave externa nos relaciona los *bookmarks* con los usuarios asociados a ellos.

4.2.4. Tabla banned_bookmarks

Esta tabla almacena todos aquellos *bookmarks* que han sido prohibidos por el/los administradores de la aplicación. Esta tabla puede contener poca o mucha información en función de la finalidad de la comunidad en la que se utilice ya que hay ocasiones en que si la comunidad es muy grande, se pueden dar casos de utilizar la aplicación para fines personales o incluir contenidos no apropiados. Por otra parte, esto nos permite realizar un control adicional de que las páginas que se marcan sigan en activo, por lo que si un administrador detecta o es informado de una página que no está activa, la puede eliminar de los registros de *bookmarks* añadiendola a esta tabla.

Las columnas de la tabla banned_bookmarks son las siguientes:

- banned_bookmark_id: integer, not null

  Este campo almacena el identificador que asigna OpenACS a cada nuevo objeto que se crea, este es único y se utiliza para mantener la integridad de la base de datos.

- bookmark: text, not null

  Este campo contiene la URL que ha sido prohibida o eliminada. Este campo siempre contiene información.

- comment: text

  En este campo se almacena cualquier comentario que el administrador haga a la hora de prohibir un determinado *bookmark*. Es recomendable indicar la razón por la que se elimina de la lista de *bookmarks* que tiene la aplicación.

- context_id *

  Este campo contiene el identificador de la instancia del paquete en la cuál se creó este objeto. Este dato nos permite diferenciar *bookmarks* prohibidos que pertenecen a distintas comunidades que usan el servicio de gestión de *bookmarks*.

La clave primaria de la tabla es banned_bookmark_id. La tabla también tiene una clave externa que relaciona banned_bookmark_id con el campo object_id de la tabla acs_objects.

4.2.5. Tabla bookmark_users

Esta tabla almacena la información de cada usuario para poder personalizar la visualización de los contenidos de la aplicación mediante los parámetros recuperados de esta tabla y que se pueden modificar en la página de configuración de usuario.

Las columnas de la tabla bookmark_users son las siguientes:

- bookmark_users_id: integer, not null

  Este campo almacena el identificador que asigna OpenACS a cada nuevo objeto que se crea, este es único y se utiliza para mantener la integridad de la base de datos.

- delicious_user: text

  Aquí se almacena el nombre de usuario de delicious de cada miembro de la comunidad que tiene cuenta en delicious.

- user_filter: text

  Aquí se almacena la información de filtrado personal de cada usuario que se realiza al mostrar los *bookmarks* contenidos en la aplicación, esto permite tener comunidades más grandes con más intereses y que sea cada usuario el que realice su propio filtrado para visualizar aquello que sea más cercano a sus necesidades o intereses.

- user_order_by: text

  Este campo contiene el tipo de clasificación que cada usuario quiere obtener al visualizar los *bookmarks*.

- creation_user *

  Este campo contiene el identificador del usuario que se relaciona con la información de la cuenta de delicious y los filtros almacenados en esta tabla. Es necesario este campo para asociar un identificador de usuario de OpenACS con el usuario de la aplicación y su correspondiente información de usuario de delicious y los filtros.

- context_id *

  Este campo contiene el identificador de la instancia del paquete en el cuál se creó el objeto. Nos permite diferenciar la información de un mismo usuario que pertenece a la vez a más de una comunidad que usa el servicio de gestión de *bookmarks*.

La clave primaria de la tabla es bookmark_users_id. La tabla también tiene una clave externa que relaciona bookmark_users_id con el campo object_id de la tabla acs_objects.

4.3. Procesado de feeds y actualización de *bookmarks*

En esta sección se va a tratar el diseño del bloque encargado de la actualización de datos en la aplicación. Esta actualización se realiza periódicamente y de forma automática en segundo plano, por lo que no se necesita controlar esta función de forma manual.

El procedimiento que se va a explicar a continuación sirve como núcleo del paquete ya que a partir de las cuentas de usuario de delicious que tiene almacenados en la base de datos, este puede obtener la información almacenada en esas cuentas y filtrarla para almacenar los *bookmarks* útiles. También se actualiza la información ya almacenada si se han producido cambios.

En el siguiente diagrama de flujo podemos ver de forma general el orden de ejecución que se sigue para la actualización de *bookmarks*:

Como se puede ver en el anterior diagrama de flujo, el procedimiento update_bookmarks obtiene los nombres de usuario en delicious de los usuarios de la aplicación de *bookmarks*, genera una dirección (URL) a partir del nombre de usuario de las preferencias que haya seleccionado el administrador de la aplicación y llama al procedimiento parse_feed que es el encargado de parsear el archivo obtenido para extraer la información útil.

En los siguientes diagramas se muestra el diagrama de flujo detallado del procedimiento parse_feed. Lo primero que se hace es obtener el archivo rss con los feeds obtenidos a partir de la URL generada en el procedimiento update_bookmarks y se genera el árbol del archivo obtenido.

En este punto está el comienzo de un bucle con el que se recorren todos los episodios del feed obtenido. Dicho bucle no ha sido mostrado en el diagrama de flujo ya que incrementaba mucho la complejidad del diagrama y se perdía una visión más enfocada a lo que pasa dentro del procedimiento que al propio flujo.

Cada iteración comienza con un nuevo elemento del *feed*, se formatea el enlace para que siempre tenga la misma forma a la hora de almacenarlos en la base de datos y se comprueba si el *bookmark* está prohibido.

En caso de que esté prohibido se pasa al siguiente elemento y se pasa al paso anterior.

Si no está en la lista de *bookmarks* prohibidos, se obtienen las categorías

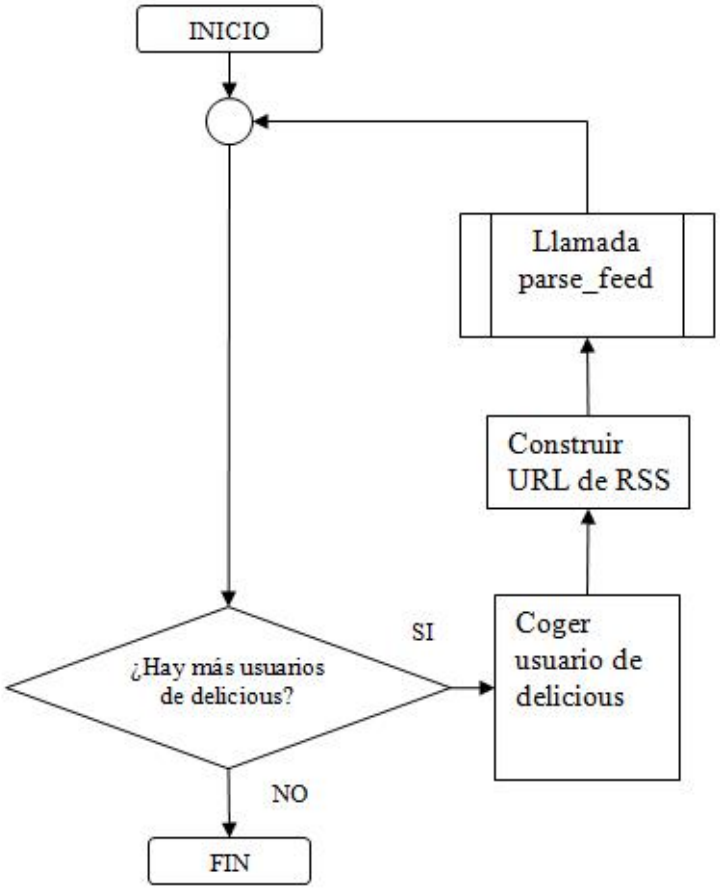

Figura 4.2: Diagrama de flujo de update_bookmarks

de dicho *bookmark* y se comprueba si el *bookmark* ya está almacenado como *bookmark* en la comunidad.

- Bookmark no existe: en este caso el flujo del procedimiento seguiría según se muestra en el diagrama de flujo en la figura 4.4

  En este caso se obtiene toda la información útil del episodio y se crean las entradas con la información obtenida en las distintas tablas de la base de datos. Con esto finaliza esta iteración del bucle.

- Bookmark sí existe: en este caso el flujo del procedimiento seguiría según se muestra en el diagrama de flujo en la figura 4.5

Primero vemos si existe un guid asociado con este *bookmark* y para el usuario del que procede el feed. Si no lo hay, se almacena como nuevo usuario que tiene este *bookmark*, se actualiza la lista de categorías por si en el episodio apareciesen nuevas categorías asociadas al *bookmark* y se acaba esta iteración del bucle.

Si ya existe un guid para este *bookmark* y este usuario, se comprueba si el que se ha recibido es el mismo que el que estaba almacenado. Si es igual significa que se ha recibido la misma información que en un feed anterior y por tanto finaliza la iteración del bucle.

Si el guid es distinto, se actualiza la fecha de publicación del *bookmark* si la del episodio es más moderna que la que está almacenada, después se actualiza el guid y se crean nuevas categorías si el episodio trae alguna que no esté ya almacenada asociada a este *bookmark*. Aquí acaba la iteración del bucle.

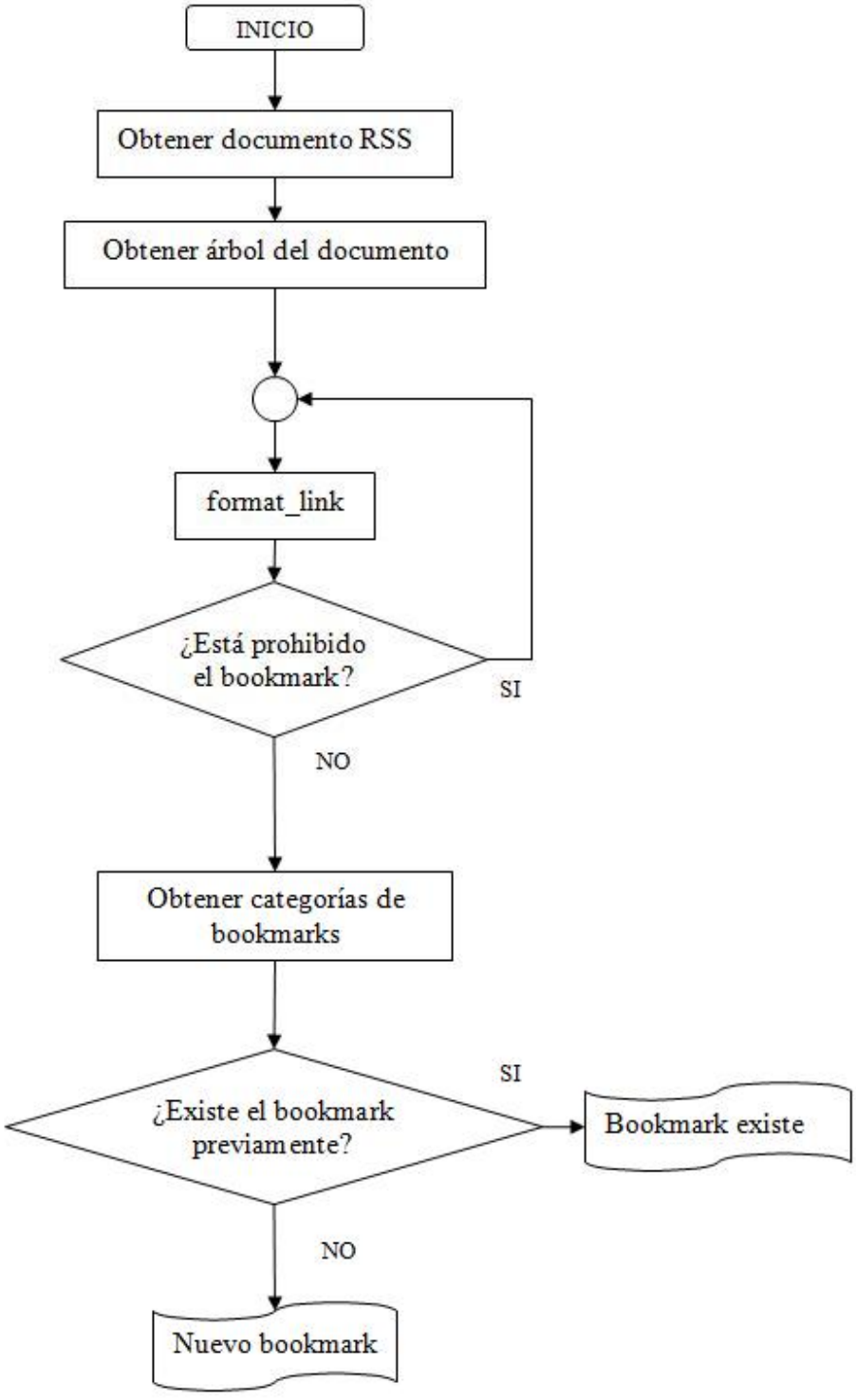

Figura 4.3: Diagrama de flujo de parse_feed: Inicio

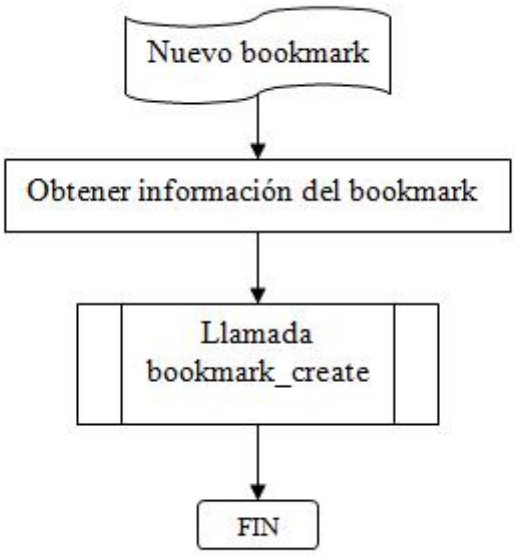

Figura 4.4: Diagrama de flujo de parse_feed: Nuevo *bookmark*

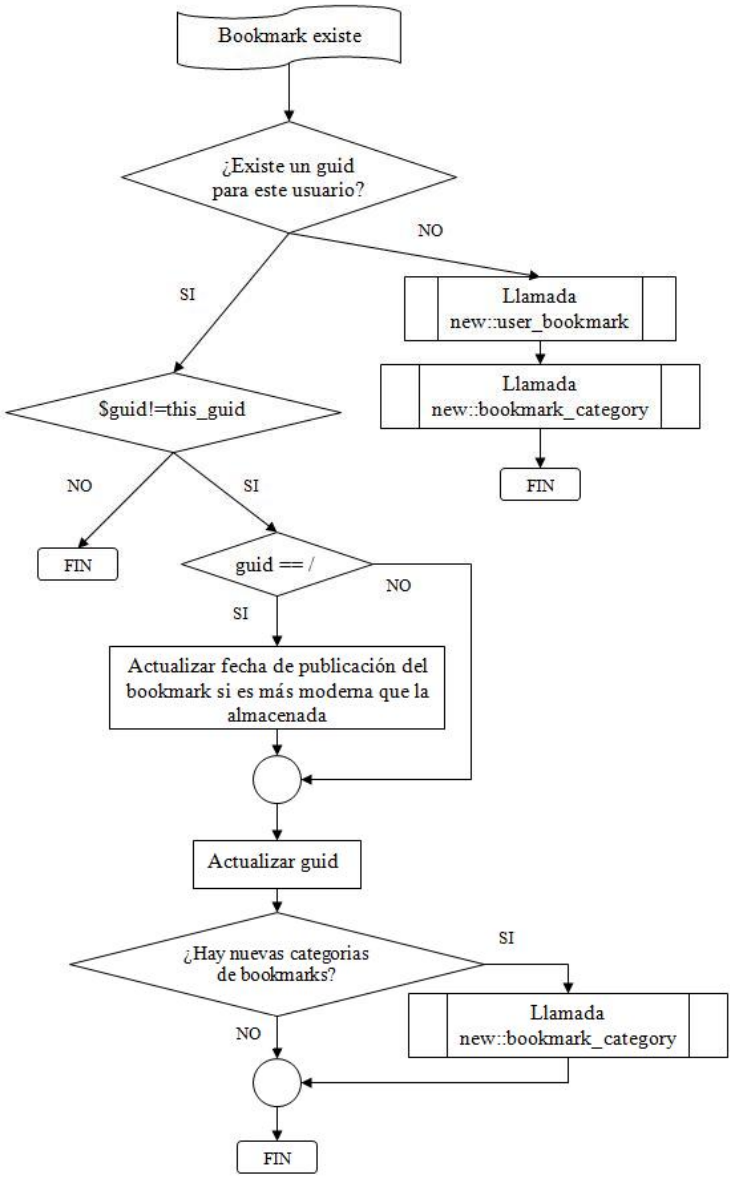

Figura 4.5: Diagrama de flujo de parse_feed: *Bookmark* existe

Capítulo 5

Paquete Rating

La implementación de la aplicación de filtrado de *bookmarks* comenzó con el diseño de la utilidad que iba a tener la aplicación y para ello se debía implementar un paquete que incluyera dicha información. Como parte de dicha utilidad, se dicidió añadir un sistema de valoraciones para obtener información más valiosa de los contenidos útiles para la comunidad. Con esta funcionalidad, se ofrecía un filtrado adicional de la información disponible utilizando la inteligencia colectiva de la comunidad; no sólo realizando un filtrado sobre los *bookmarks* de cada uno de los miembros, sino que además da la oportunidad de ofrecer una realimentación por parte de los propios usuarios para clasificar los resultados del filtrado basado en las experiencias de cada uno de los miembros de la comunidad.

Por las razones expuestas anteriormente, se procedió al diseño de un paquete adicional llamado Rating que ofrecería este servicio de forma externa. De esta manera se podría reutilizar en otros paquetes e incrementar la funcionalidad del propio paquete de forma independiente.

5.1. Funcionalidad ofrecida por el paquete Rating

El concepto fundamental sobre el que se sustenta la creación del paquete Rating es dar la posibilidad de evaluar objetos en OpenACS/.LRN. Por esta razón, este paquete ofrece un api que nos da los procedimientos necesarios para mostrar los resultados de las evaluaciones de los objetos y evaluar objetos siguiendo distintas formas de evaluación (tick o cruz, valoración de 1 a 5, valoración de 1 a 10, pulgar hacia arriba o hacia abajo, estrellas, etc).

Adicionalmente, para dar mayor valor al paquete desarrollado, se estudiaron funcionalidades extras que podrían agregarse al paquete a partir de ideas

obtenidas de otras plataformas y servicios y adaptándolos a la evaluación de objetos.

Primero se añadió un sistema para almacenar comentarios sobre el objeto evaluado, de esta forma se ofrecía un servicio con interactividad incrementada dentro de la comunidad al poder emplazar discusiones específicas a objetos determinados en lugar de utilizar otros servicios como pueden ser foros, chats o listas de correo. Este sistema de comentarios asociados a cada objeto lo podemos ver en los blogs, aunque la finalidad es diferente ya que en los blogs se utilizan los comentarios como realimentación de cada artículo escrito en el blog mientras que la finalidad de los comentarios dentro del paquete rating es poder tener asociada la discusión sobre algo específico a un determinado concepto u objeto asociado directamente a ese concepto u objeto. De esta manera se aprovechan las ventajas de tener una mayor organización de la información, ya que en todo momento se puede acceder a todo lo escrito anteriormente sobre un determinado objeto directamente.

También se añadió la posibilidad de que cada usuario añadiera un bloque de texto llamado descripción que fue concebido para que cada usuario ofreciera su propia explicación o descripción del objeto evaluado, pero que puede ser usado de otras maneras como pueda ser para posicionarse con una posición a favor o en contra y como esta funcionalidad va provista de un sistema de votación, se pueden obtener las descripciones más significativas desde el punto de vista de los miembros de una comunidad.

Hay que destacar que tanto la funcionalidad de comentarios como la de descripciones son opcionales y pueden ser usadas o no como parte de la evaluación de los objetos de otro paquete en función de las necesidades del desarrollador.

5.2. Modelo relacional de la base de datos

Como ya se ha comentado antes, el paquete Rating se ha diseñado para que pueda ser utilizado por otras aplicaciones que se desarrollen en el futuro o actualizaciones de aplicaciones existentes. Por esta razón, el paquete Rating ofrece un modelo relacional sencillo a la vez que potente para tratar con diferentes casos de uso y permitiendo incluir nuevos casos de uso sin necesidad de modificar el modelo relacional del paquete.

Algunos campos de las tablas viene marcados con un asterisco (*), esto significa que dicho campo no se encuentra en dicha tabla pero su información forma parte de dicha tabla y no se ha incluído en ella por estar almacenado en la tabla acs_objects donde se crea una nueva entrada cada vez que se crea un nuevo objeto ya que esta tabla forma parte de la arquitectura de la

plataforma OpenACS y contiene, entre otros campos, el identificador único que OpenACS asigna a cada objeto creado, pero también información sobre el paquete en el que se creó el objeto, contexto, identificador del usuario que creó el objeto, fecha de creación, nombre del objeto y otros campos que también son de utilidad para la plataforma.

En la figura 5.1 se muestra un diagrama con el modelo relacional del paquete *Rating*. Después se detalla el contenido de cada una de las tablas explicando su uso dentro del paquete.

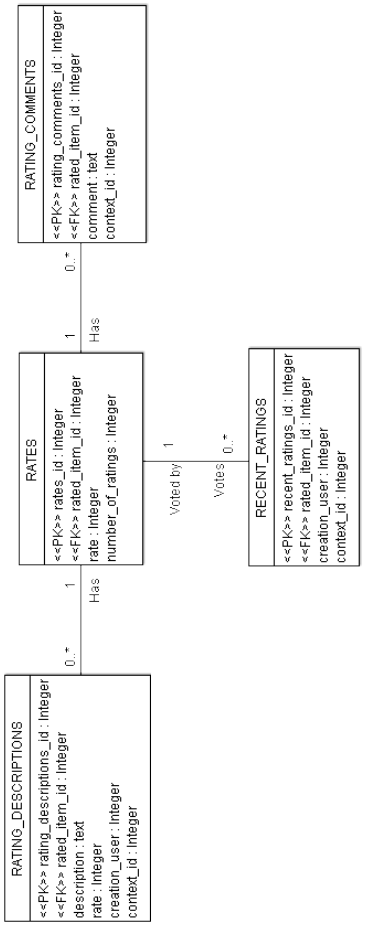

Figura 5.1: Modelo relacional del paquete Rating

5.2.1. Tabla Rates

Esta tabla es la más importante del paquete ya que contiene la columna rated_item_id que contiene la información del item_id u object_id con la que podemos asociar un determindado objeto con su información de valoración, así como de comentarios y descripciones de dicho objeto.

Las columnas de la tabla Rates son las siguientes:

- rates_id: integer, not null

  Este campo almacena el identificador que asigna OpenACS a cada nuevo objeto que se crea, este es único y se utiliza para mantener la integridad de la base de datos.

- rated_item_id: integer

  Este campo guarda el identificador único que OpenACS dió al objeto evaluado en el momento de su creación. Este campo es el que conecta el servicio que ofrece el paquete Rating con el exterior.

- rate: double, precision

  Guarda el valor de la valoración. Puesto que este paquete es de uso generalista, rate guarda la suma de todas las valoraciones que se hayan hecho del paquete, para luego tratar dicho dato en función de la valoración elegida en cada caso. Para que este método funcione correctamente, también se debe tener la información del campo number_of_ratings.

- number_of_ratings: integer

  Almacena el número de veces que se ha evaluado el objeto. Esta información es útil para poder calcular la valoración media en función del tipo de evaluación que se utilice.

La clave primaria de la tabla es rates_id y tiene una restricción: rated_item_id debe ser único. La tabla también tiene dos claves externas. La primera une rated_item_id con el campo object_id de la tabla acs_objects y la segunda une rates_id con el mismo campo de la tabla acs_objects.

5.2.2. Tabla recent_ratings

Esta tabla sirve para limitar las votaciones de los usuarios. Cuando un usuario vota un determinado objeto se añade una entrada en esta tabla indicando que dicho usuario ha votado ese objeto. De esta forma, se evita que el

usuario pueda votar un número indefinido de veces modificando el resultado de la votación.

Las columnas de la tabla recent_ratings son las siguientes:

- recent_ratings_id: integer, not null

  Este campo almacena el identificador que asigna OpenACS a cada nuevo objeto que se crea, éste es único y se utiliza para mantener la integridad de la base de datos.

- rated_item_id: integer

  Este campo guarda el identificador único que OpenACS dió al objeto evaluado en el momento de su creación. Este campo es el que conecta el servicio que ofrece el paquete Rating con el exterior.

- creation_user *

  Este campo contiene el identificador del usuario que realizó la votación. Se necesita almacenar esta información para poder prohibir que dicho usuario vuelva a votar el mismo objeto.

- context_id *

  Este campo contiene el identificador de la instancia del paquete en el cuál se creó el objeto. Este dato nos permite diferenciar votaciones de un mismo usuario a un mismo objeto en distintas instancias. En la aplicación desarrollada en este proyecto no puede darse este caso, pero puesto que el paquete es de carácter generalista, se ha contemplado este caso.

La clave primaria de la tabla es recent_ratings_id. La tabla también tiene una clave externa, que relaciona recent_ratings_id con el campo object_id de la tabla acs_objects.

5.2.3. Tabla rating_comments

Esta tabla almacena todos los comentarios que los usuarios escriben en los objetos evaluados. Este es uno de los puntos críticos del paquete en cuanto a recursos se refiere ya que si las comunidades que usan este paquete son grandes, el tamaño de la base de datos puede crecer rápidamente. Por esta razón, esta tabla tiene únicamente los campos imprescindibles para un buen funcionamiento.

Las columnas de la tabla rating_comments son las siguientes:

- rating_comments_id: integer, not null

  Este campo almacena el identificador que asigna OpenACS a cada nuevo objeto que se crea, éste es único y se utiliza para mantener la integridad de la base de datos.

- rated_item_id: integer

  Este campo guarda el identificador único que OpenACS dió al objeto evaluado en el momento de su creación. Este campo es el que conecta el servicio que ofrece el paquete Rating con el exterior.

- comment: text

  En este campo se almacena el contenido del comentario. El tamaño de esta campo no está limitado.

- context_id *

  Este campo contiene el identificador de la instancia del paquete en el cuál se creó el objeto. Este dato nos permite diferenciar comentarios de un mismo objeto en distintas instancias. En la aplicación desarrollada en este proyecto no puede darse este caso, pero puesto que el paquete es de carácter generalista, se ha contemplado este caso.

La clave primaria de la tabla es rating_comments_id. La tabla también tiene una clave externa, que relaciona rating_comments_id con el campo object_id de la tabla acs_objects.

5.2.4. Tabla rating_descriptions

Esta tabla almacena las descripciones que cada usuario puede escribir sobre el objeto evaluado. Puede parecer que esta tabla es innecesaria teniendo la tabla rating_comments, pero la realidad es que ambas tablas almacenan contenidos diferentes tanto desde el punto de vista funcional como conceptual.

El concepto de descripción se utiliza como bloque de texto único que el usuario posee para dar explicaciones, asumir posiciones ideológicas o cualquier otra finalidad que se pueda acoplar a las características de lo que se llaman descripciones en este paquete. Dichas descripciones pueden ser editadas por el usuario que las creó siempre que lo crea necesario y pueden ser votadas, lo que ofrece la posibilidad de dar preferencia a ciertas descripciones frente a otras. Por estas razones, la tabla rating_descriptions contiene las siguientes columnas:

- rating_descriptions_id: integer, not null

  Este campo almacena el identificador que asigna OpenACS a cada nuevo objeto que se crea, éste es único y se utiliza para mantener la integridad de la base de datos.

- rated_item_id: integer

  Este campo guarda el identificador único que OpenACS dió al objeto evaluado en el momento de su creación. Este campo es el que conecta el servicio que ofrece el paquete Rating con el exterior.

- description: text

  En este campo se almacena el contenido de la descripción. El tamaño de esta campo no está limitado.

- rate: integer

  Este campo almacena la valoración de la descripción, cuanto mayor sea, mayor preferencia se dará a la descripción a la hora de mostrarla a los usuarios.

- creation_user *

  Este campo almacena el identificador del usuario que ha escrito la descripción. Con este valor la aplicación puede dar permisos de escritura en la descripción para las sucesivas ocasiones que dicho usuario entre en la página de descripciones de ese objeto.

- context_id *

  Este campo contiene el identificador de la instancia del paquete en el cuál se creó el objeto. Este dato nos permite diferenciar comentarios de un mismo objeto en distintas instancias. En la aplicación desarrollada en este proyecto no puede darse este caso, pero puesto que el paquete es de carácter generalista, se ha contemplado este caso.

La clave primaria de la tabla es rating_descriptions_id. La tabla también tiene una clave externa, que relaciona rating_descriptions_id con el campo object_id de la tabla acs_objects.

5.3. Integración de Rating

El paquete Rating esta diseñado para que sea fácilmente integrable en otras aplicaciones. Para ello ofrece una serie de procedimientos denominados

templates o plantillas que ofrecen la funcionalidad necesaria para ofrecer un servicio de evaluación de objetos. Estos procedimientos muestran los distintos tipos de evaluaciones que se pueden hacer como puntuaciones de 1 a 10 o puntuar como bueno o malo. Además, estos templates también devuelven toda la información útil almacenada en la evaluación de este objeto.

Con estos procedimientos se simplifica el uso del sistema de evaluaciones en otros servicios. A continuación se muestran dos ejemplos del uso de los procedimientos ofrecidos por Rating.

En la figura 5.2 vemos como se evalúan los *bookmarks* en la aplicación. El tick y la cruz se muestran llamando a rating::template::tick_or_cross, la puntuación se recupera llamando al procedimiento rating::rate::get_rate y el número de puntuaciones con rating::rate::get_number_of_ratings. Para utilizar otro tipo de evaluación se debe llamar al procedimiento que nos devuelva el modelo de evaluación que se desea: con estrellas (rating::template::stars) y de cero a diez (rating::template::zero_to_ten).

Figura 5.2: Evaluación tick y cross

Para crear otro sistema de evaluación sólo hay que crear un nuevo procedimiento que nos devuelva el interfaz gráfico del sistema de votación de modo que se pueda incluir dentro de una celda dentro de una tabla en una página o en cualquier otra parte del código de una página. También hay que tener en cuenta la lógica que sigue el sistema de votación ya que el paquete ofrece un sistema que calcula la media de las notas pero si quisiesemos otro tipo de resultado, este también debería implementarse.

Otro ejemplo de integración de la evaluación está integrado en el propio paquete Rating y se utiliza para evaluar descripciones. En la figura 5.3 se puede ver que se trata de un sistema que otorga estrellas. Si la descripción es buena se puede evaluar positivamente pulsando en la estrella amarilla y si la evaluación es negativa, se pulsa la estrella blanca. Este sistema no tiene memoria por lo que si se ha votado n veces positivamente y otras n veces negativamente, entonces la descripción tiene una evaluación de 0 igual que otra descripción que no se ha evaluado nunca.

Para que la integración de la funcionalidad de Rating sea completa, se ofrecen dos páginas para mostrar tanto las descripciones como los comen-

Figura 5.3: Evaluación con estrellas

tarios sobre el objeto evaluado y para poder integrar dichas páginas en otra aplicación únicamente hay que incluir un enlace a dichas páginas en el punto que nos interese.

En la figura 5.4 podemos ver como se integra la página de comentarios en la página principal de la aplicación de gestión de *bookmarks* y en la figura 5.5 se muestran los enlaces a las páginas de comentarios y descripciones dentro de la página de información de un determinado *bookmark*.

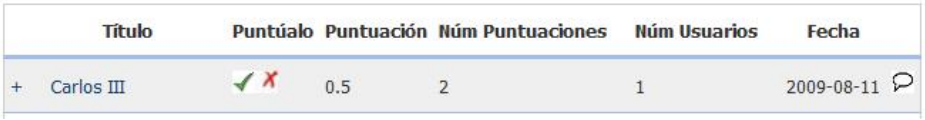

Figura 5.4: Icono de comentarios integrado en Bookmark

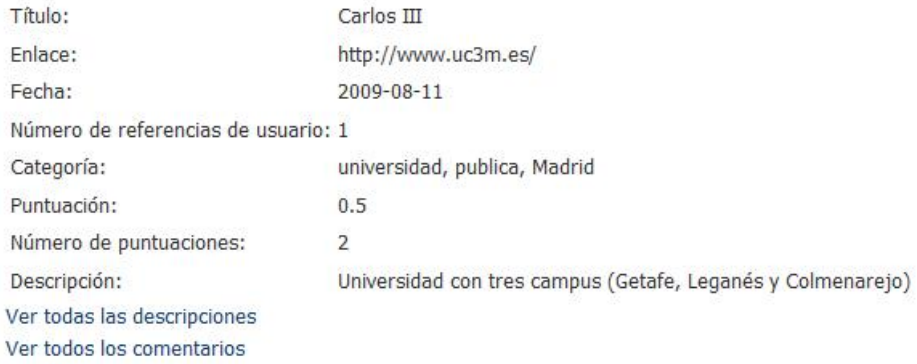

Figura 5.5: Enlaces a comentarios y descripciones

Capítulo 6

Pruebas

En este capítulo se describirá la estrategia utilizada para la realización de las pruebas, así como el tipo de pruebas realizado. Esta tarea es muy importante dentro de la realización del proyecto ya que sirve para validar la funcionalidad descrita en los capítulos anteriores.

Se diseñaron dos escenarios de pruebas, el primero durante el desarrollo de los distintos bloques que se fueron desarrollando y otro para probar la aplicación finalizada. El primer escenario se destribuyó en el tiempo durante todo el periodo de desarrollo, ya que se decidió que era mejor resolver pequeños errores a medida que se iban detectando con las pruebas antes que esperar a desarrollar un bloque completo y luego encontrarse con un grupo de errores que sean más difíciles de resolver que si se hubiesen resuelto por separado durante el desarrollo. El segundo escenario comprendía la certificación de la funcionalidad de la aplicación completa incluyendo pruebas de integración y validación.

Hay que tener en cuenta que en desarrollos software no se puede obtener certeza absoluta de que la aplicación funcione correctamente el 100 % de los casos, por esta razón existe un riesgo de que algún *bug* se haya quedado sin detectar y por tanto produzca un comportamiento indeseado de la aplicación. Si se diese este caso, se tendría que poner en contacto con el autor de este proyecto para solucionar dicho error.

6.1. Escenario de desarrollo

Durante el desarrollo de los distintos bloques funcionales que componen la aplicación de gestión de *bookmarks* se realizaron baterías de pruebas que tenían como objetivo la verificación del correcto funcionamiento del código escrito. En esta fase de pruebas se realizaron baterías lo más exhaustivas

posible ya que si se localizan los errores en las primeras fases de desarrollo, el coste de arreglar dichos errores es mucho menor que si estos se descubren cuando uno se encuentra en la fase de integración o de validación.

Se aplicaron las llamadas pruebas de caja blanca en las que no sólo se comprueba que la funcionalidad de los distintos bloques de código es correcta sino también que no ocurra nada inesperado durante la ejecución de dicho bloque de código. Para ello se comprobó que:

- Bloques if-then-else funcionen correctamente, se asignen correctamente los valores que se comparan, estén bien descritas las condiciones y se ejecute el bloque deseado en función del resultado de la condición.
- Bucles se ejecuten correctamente con condiciones de continuación y finalización correctas.
- Inicialización de variables se realiza correctamente
- No haya bloques de código que no se ejecuten nunca porque las condiciones no se cumplan nunca.
- Datos se almacenen en lugar correcto, sin alteraciones no deseadas y que se generen las referencias necesarias para recuperar la información correctamente.
- LLamadas a la base de datos se realizan correctamente (se recupera únicamente la información deseada y se actualiza la información correctamente).
- Páginas se redirijan correctamente, sin brechas de seguridad (es decir, que haya usuarios sin permiso que accedan a la aplicación y que usuarios que no sean administradores puedan acceder a las funciones de administración de la aplicación).

6.2. Escenario de integración y validación

Para finalizar el desarrollo de un proyecto software se deben llevar a cabo pruebas de integración y de validación de la aplicación para ofrecer fiabilidad al cliente o al usuario de la aplicación. Esta fase requiere un gran esfuerzo debido a que es el punto previo a sacar el producto al público, por esta razón se debe tratar de validar todo lo que sea posible.

En este proyecto se ha dedicado parte del tiempo a verificar la conformidad de la aplicación con las especificaciones que se describieron en la fase de

diseño del gestor de *bookmarks*. También se ha tenido un especial cuidado en el proceso de validación, realizando pruebas de uso de la aplicación para comprobar que esta funcionaba correctamente.

Pruebas realizadas para la validación de la aplicación:

- Pruebas realizadas sobre la parte de la aplicación integrada en el paquete Bookmark:
  - Página de registro aparece únicamente cuando un usuario accede por primera vez a la aplicación. Se comprueba que los datos obligatorios son introducidos y los opcionales no son obligatorios, se crea el nuevo usuario en la base de datos y se redirige a la página de inicio.
  - Aparecen las funciones de administración únicamente para usuarios con privilegios de administración, se modifican los datos para la instancia en la que se encuentra el administrador, se muestran los *bookmarks* baneados y se gestionan *bookmarks* baneados correctamente.
  - Página de configuración de usuario se muestra correctamente y se actualizan los datos modificados por el usuario.
  - Se actualizan los *bookmarks* almacenados en la aplicación periódicamente.
  - Se pueden añadir nuevos *bookmarks* manualmente y no se permite que se añadan cuando estos existían previamente.
- Pruebas realizadas sobre la parte de la aplicación integrada en el paquete Rating:
  - Se almacenan las votaciones y se prohibe repetir votación del mismo objeto para un mismo usuario.
  - Página de descripciones y de comentarios se muestran correctamente, ofreciendo únicamente la información relativa al objeto evaluado.
  - Se añaden comentarios y descripciones asociadas al usuario que las escribe y en el caso de descripciones se actualiza su contenido si este lo modifica.
  - API para mostrar los distintos sistemas de votaciones está completo y no da errores.

- Funcionalidad para evitar votaciones múltiples por parte de un usuario a un objeto funciona correctamente y se borran los logs periódicamente.

Las pruebas mencionadas anteriormente tenían como objetivo comprobar que funcionaba correctamente toda la funcionalidad integrada en la aplicación de gestión de *bookmarks*. Adicionalmente, también se comprobó el paquete Rating para que la funcionalidad adicional ofrecida por dicho paquete no tuviese errores.

Capítulo 7

Historia del proyecto

En este capítulo se comentará la evolución temporal del proyecto, incluyendo la estrategia seguida para la consecución de hitos. También se incluye un presupuesto de la realización del proyecto adaptado a los precios de mercado.

7.1. Evolución temporal

A continuación se muestran las tareas realizadas durante el desarrollo del proyecto. También se incluye un diagrama de Gantt que se muestra en la figura 7.1 en la en la última página de este capítulo.

- 29/09/2008-26/10/2008: Estudio de la plataforma OpenACS/.LRN, subversión y máquina virtual.
- 27/10/2008-30/10/2008: Análisis de necesidades de la aplicación a desarrollar y especificación de los mismos.
- 31/10/2008-10/11/2008: Diseño de la aplicación.
- 11/11/2008-11/11/2008: Especificación funcionalidad primer prototipo.
- 12/11/2008-16/12/2008: Desarrollo del primer prototipo.
- 17/12/2008-23/12/2008: Pruebas primer prototipo.
- 24/12/2008-1/01/2009: Vacaciones Navidad.
- 2/01/2009-5/01/2009: Especificación funcionalidad segundo prototipo.

- 6/01/2009-25/02/2009: Desarrollo del segundo prototipo.
- 26/02/2009-8/03/2009: Pruebas segundo prototipo.
- 9/03/2009-10/03/2009: Especificación funcionalidad prototipo final.
- 11/03/2009-21/04/2009: Desarrollo del prototipo final.
- 22/04/2009-30/04/2009: Pruebas prototipo final.
- 6/04/2009-8/06/2009: Escribir memoria del proyecto.
- 1/05/2009-5/06/2009: Pruebas de integración.
- 9/06/2009-12/06/2009: Revisión definitiva memoria.

7.2. Presupuesto

El presupuesto del proyecto se divide en dos grupos: costes de personal y costes de material. El presupuesto se ha minimizado lo máximo posible debido al uso de software libre, por esta razón el proyecto es lo más competitivo posible. A continuación se detallan todos los costes de material asumidos para la realización del proyecto:

- Ordenador personal (PC):

  Equipo fundamental para el desarrollo de un proyecto software. El equipo utilizado es un Intel(R) Core(TM) 2 Duo E7300 a 2.66GHz, 4GB de RAM, 150GB de disco duro, pantalla LCD de 19".

  Coste: 531€(PC)+209€(monitor) = 740€

- Máquina virtual con dotlrn:

  Software de virtualización VMware y máquina virtual .LRN Virtual Machine descargable de la web del departamento de telemática de la Universidad Carlos III[1].

  Coste = 0€

- Controlador de versiones:

  Subversion[2] (SVN)

  Coste = 0€

[1] https://gradient.it.uc3m.es/xowiki/dotlrn_vm
[2] http://subversion.tigris.org

- Editor de LaTeX:

  Miktex 2.7[3] y TeXnicCenter[4]

  Coste = 0€

A continuación se detalla el coste de personal asociado al proyecto:

- 1 Ingeniero de Telecomunicación:

  Sueldo anual: 46000€por 1760 horas (datos obtenidos del Colegio Oficial de Ingenieros de Telecomunicaciones[5]). Sueldo = 26,14€/hora

  Coste = 640horas x 26.14€/hora = 16729,6€

Costes generales	
Ordenador personal	740 €
Software desarrollo y edición	0 €
Costes de personal	
26,14€/hora × 640 horas	16729,6 €
Total	17469,6 €

Tabla 7.1: Presupuesto del proyecto

[3]http://miktex.org
[4]http://www.texniccenter.org
[5]www.coit.es

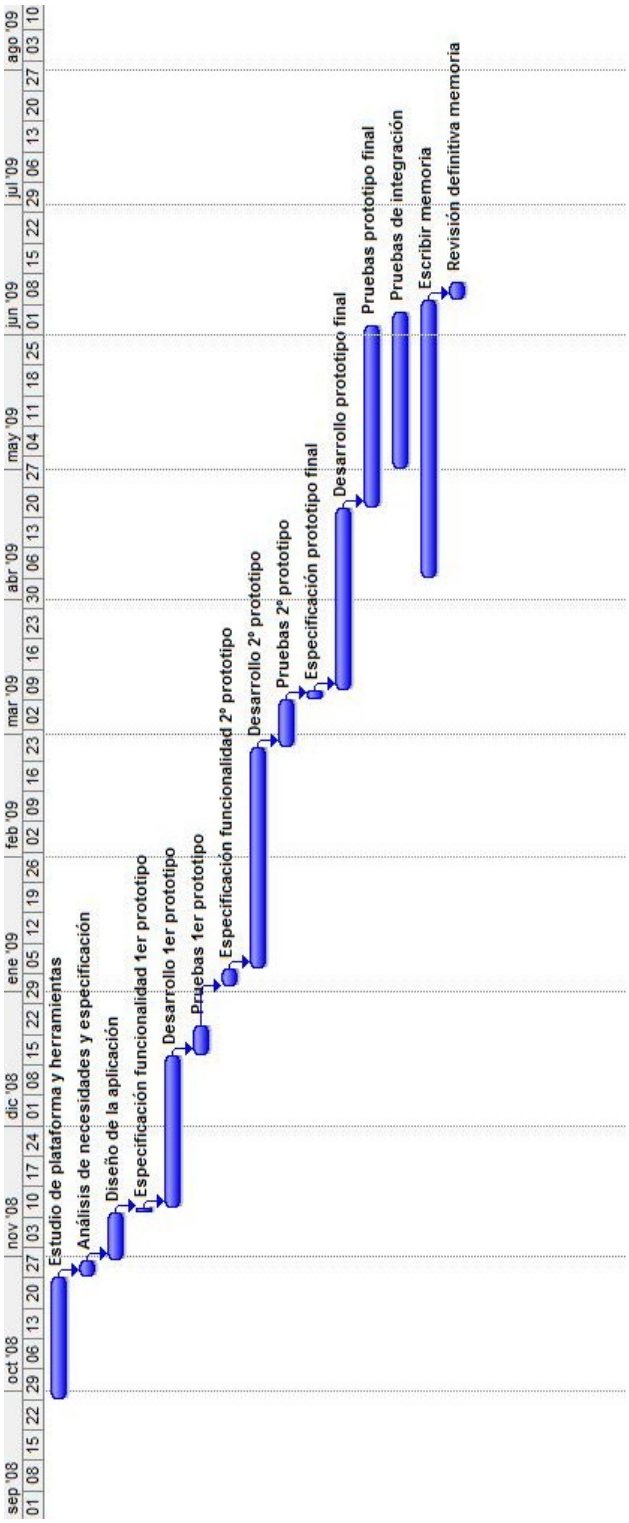

Figura 7.1: Diagrama de Gantt

Capítulo 8

Conclusiones y trabajos futuros

Por último, en este capítulo se expondrá una conclusión respecto al trabajo realizado y las líneas futuras que se pueden seguir para mejorar la aplicación e implementar nuevas funcionalidades.

Primero hay que destacar que se han conseguido los objetivos perseguidos en el proyecto. Primero se ha estudiado una plataforma de teleeducación (.LRN), incluyendo su arquitectura, la plataforma OpenACS sobre la que está desplegado .LRN, el servidor AOLserver y la base de datos PostgreSQL. También se ha aprendido un nuevo lenguaje, TCL, que por ser interpretado tiene bastantes diferencias con los compilados (Java, C, etc).

También se ha estudiado la integración de servicios externos en una aplicación y se ha desarrollado una aplicación que utiliza los conocimientos adquiridos para obtener la información del servicio delicious para los usuarios miembros de una comunidad y posteriormente tratar dicha información.

Por último, también se ha obtenido los conocimientos necesarios para la utilización de máquinas virtuales, las cuales son de gran utilidad; un gestor de incidencias, cuya utilidad es directamente proporcional al tamaño del proyecto; depósitos de archivos para el almacenamiento del trabajo generado y los controladores de versiones que aunque hay que sufrir un proceso inicial para familiarizarse con ellos, luego resultan de gran utilidad para poder solucionar errores y recuperar estados anteriores del desarrollo de un proyecto de software.

Como valor añadido, no incluído en los objetivos del proyecto, también se ha aprendido LaTeX para la redacción de esta memoria. Se ha necesitado un proceso de aprendizaje para la utilización de este lenguaje pero ha resultado de gran utilidad y seguro que también lo será en el futuro.

Para continuar con la línea de investigación y el trabajo realizado en este proyecto se pueden identificar las siguientes líneas futuras:

1. Futuros trabajos sobre la aplicación desarrollada:

- Ampliación de la interacción con delicious

  Se han obtenido resultados satisfactorios acordes a lo que se buscaba en este proyecto, pero se puede incrementar la interacciòn con delicious accediendo a la información sobre *bookmarks* privados que pueda tener cada usuario. Este caso no se ha contemplado en el diseño de la aplicación ya que si se quieren compartir dichos *bookmarks* es lógico que estén como públicos, pero puede haber comunidades en las que también se quieren mostrar los privados.

- Integración de otros servicios de marcadores como fuentes de información

  La aplicación desarrollada es de gran utilidad por conglomerar la información relativa a fuentes de información que sean de interés común para el conjunto de miembros de una determinada comunidad. Por esta razón, y para crear una aplicación lo más usable posible sin necesidad de que los usuarios deban utilizar nuevos servicios, se decidió utilizar como servicio de almacenamiento de *bookmarks* delicious. También se ha dado la opción de añadir *bookmarks* manualmente, ya que de esta forma se permite que usuarios que no tengan cuenta en delicious puedan agregar sus propios *bookmarks* relevantes sin la necesidad de abrirse una cuenta en delicious. Sin embargo, puesto que existen numerosos sistemas de gestión de *bookmarks* online, sería conveniente incluir otros servicios además del de delicious ya que el usuario puede no tener delicious pero si otro.

- Creación de sistema de control de nuevos *bookmarks*

  Para reducir la necesidad de supervisión por parte de los administradores del contenido que se almacena en la aplicación, se puede crear un sistema automático de protección que analice el *bookmark* añadido descargándose la página enlazada en búsqueda de metatags determinados así como la búsqueda de palabras clave en el contenido de dicha página (aplicando minería de datos).

2. Trabajos independientes de la aplicación desarrollada:

   - Creación de una guía de desarrollo de aplicaciones para OpenACS/.LRN

     Una parte fundamental del desarrollo del proyecto ha sido el aprendizaje de la plataforma OpenACS/.LRN puesto que no la conocía con anterioridad y, a parte de desarrollar la aplicación expuesta en esta memoria, también he aprendido una metodología

de diseño de aplicaciones que podría ser de utilidad para nuevos desarrolladores. Por esta razón, es importante esta línea futura de trabajo ya que facilitaría el desarrollo de aplicaciones.

- Utilización de la aplicación desarrollada como plantilla para nuevas aplicaciones

  La aplicación desarrollada en el proyecto tiene una finalidad específica, la gestión de *bookmarks* para una comunidad en función; sin embargo, la arquitectura de la aplicación es la misma que la que usaríamos para realizar otras aplicaciones que se utilicen con elementos que puedan ser etiquetables. Ejemplo: aplicación de compartición de fotografías que sería equivalente a flickr[1] pero utilizando únicamente fotografías de los miembros de la comunidad. Otros ejemplos podrían ser una aplicación para buscar un restaurante para cenar (restaurantes etiquetados por precios, tipo de comida y evaluados por los usuarios) o la creación de un foro (no como los que conocemos actualmente que están organizados por categorías sino que cada hilo tuviese unas etiquetas de modo que realicemos búsquedas sobre dichas etiquetas de tal forma no habría duplicidad de hilos de discusión ni tampoco hilos colocados incorrectamente).

[1] http://www.flickr.com/

Apéndice A

Manual de instalación

La instalación de la aplicación se realiza en dos fases. En la primera fase se instala el paquete Rating que ofrece el servicio de evaluación de objetos. Una vez instalado este paquete hay que instalar el paquete Bookmark. La instalación se tiene que realizar en este orden ya que Bookmark utiliza el servicio ofrecido por Rating y no permite que se instale este paquete si no está instalado Rating.

Este manual de instalación parte del estado en el que la instancia de OpenACS/.LRN ya ha sido instalada. Si desea instalar una instancia de OpenACS/.LRN es recomendable seguir el manual que puede encontrar en la página de la comunidad de OpenACS[1].

Para comenzar con la instalación hay que entrar en la plataforma como administrador. Una vez introducidas las credenciales hay que pulsar en la pestaña de Administración/Administration y pinchar en el enlace a Administración de Sitio Principal/Site Wide Administration (ver figura A.1).

Aparecerá una nueva página como la mostrada en la figura A.2. Aquí tiene que pulsar el botón Developer's Admin.

En la página de Administración de desarrolladores pulsaremos en el enlace de Package Manager (ver figura A.3). La página de Package Manager que verá será similar a la de las figuras A.4 y A.5. En la parte inferior de esta página encontramos un enlace denominado Install Packages.

En la página de Package Installation encontramos una tabla con todos los paquetes que tenemos en la instancia pero no están instalados (ver figura A.6). Seleccionamos el paquete Rating y pulsamos en Siguiente/Next (ver figura A.7).

Aparecerá una nueva página (A.8) de instalación de paquetes en la que nos muestra un archivo para la instalación del modelo de datos. Marcaremos

[1] http://www.openacs.org/doc/acs-admin.html

la casilla en caso de que no esté marcada y pulsamos el botón Install Packages. La instalación comenzará y veremos una página como la mostrada en A.9. Si la instalación se completa satisfactoriamente, la parte de abajo de la página que se muestra será como la de la figura A.10. El siguiente paso será reiniciar el servidor para que se actualicen los cambios tras la instalación del paquete Rating.

Una vez reiniciado el servidor, volvemos a la página Package Installation siguiendo los pasos explicados anteriormente. Buscamos el paquete Bookmark, lo seleccionamos y pulsamos en Next para proceder a su instalación. En la página que nos aparece como la de la figura A.12 marcamos la casilla con el archivo que nos dice que es del tipo Data Model Installation y donde pone Mount Package under the main site lo marcamos si queremos crear una instancia del servicio de *bookmarks* que aparecerá en la dirección que indiquemos en el cuadro de texto. Si no queremos crear una instancia del servicio no marcamos esa casilla y se creará posteriormente cuando creemos una comunidad. Pulsamos en Install Packages y si la instalación es correcta se mostrará una página como la de las figuras A.13 y A.14.

proyecto/instalacion1.jpg

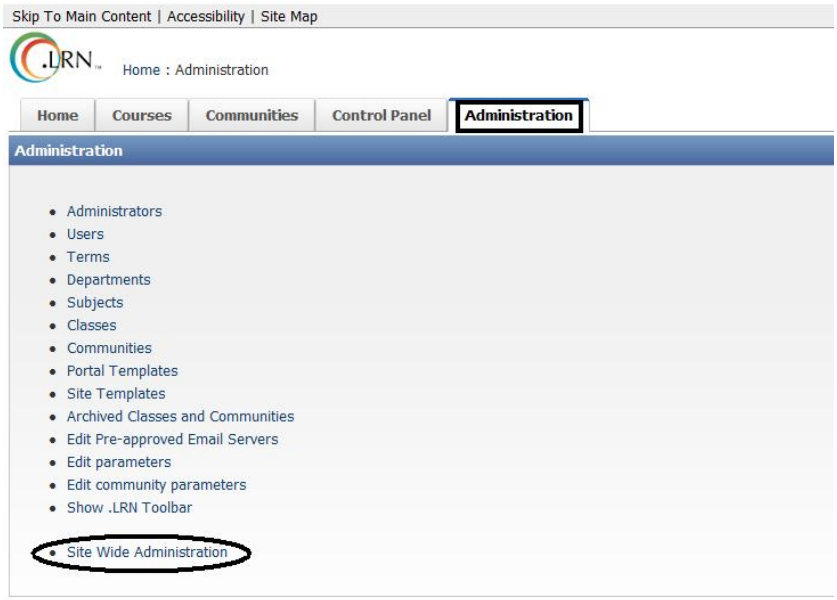

Figura A.1: Administración de OpenACS

proyecto/instalacion2.jpg

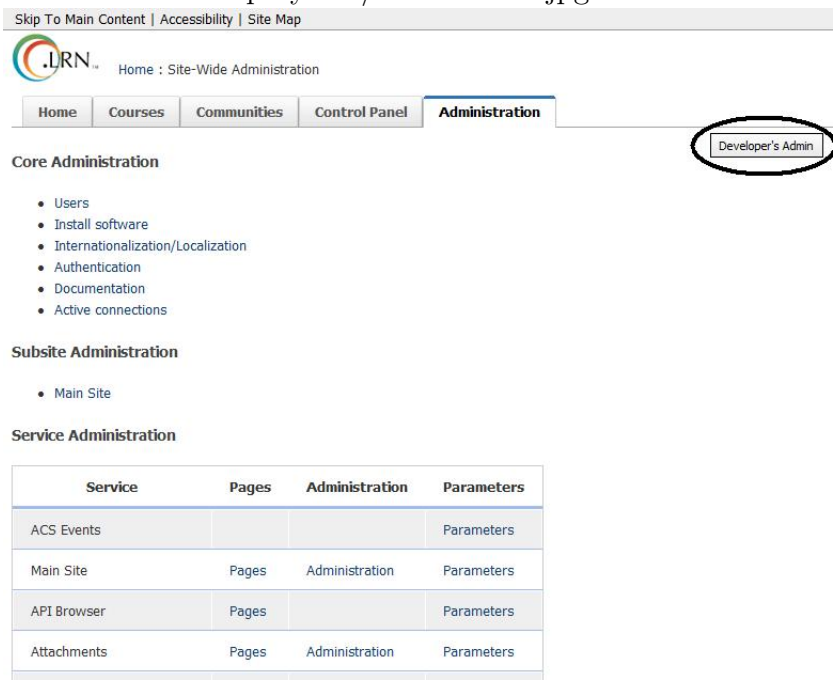

Figura A.2: Administración de sitio principal

proyecto/instalacion3.jpg

Figura A.3: Administración de desarrollador

proyecto/instalacion4.jpg

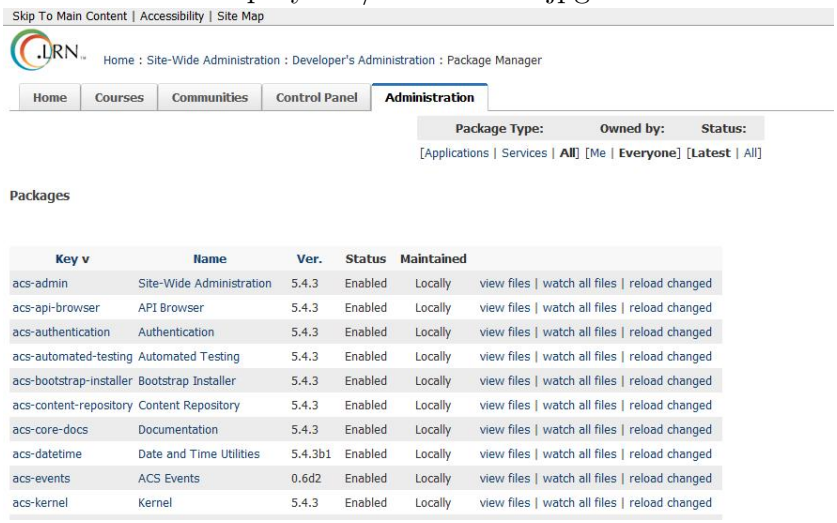

Figura A.4: Página de Package Manager (principio)

proyecto/instalacion5.jpg

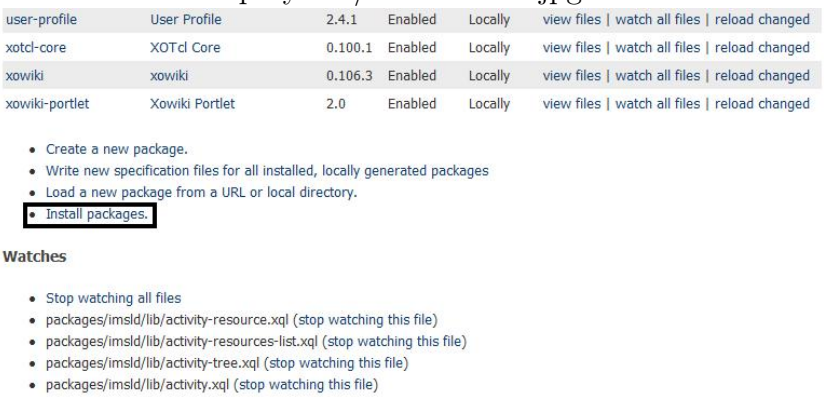

Figura A.5: Página de Package Manager (final)

proyecto/instalacion6.jpg

Package Installation

Main Site : Site-Wide Administration : Package Manager : Package Installation

Please wait while the installer searches your system for packages to install ...

Done.

Select Packages to Install

Please select the set of packages you'd like to install.

uncheck all boxes | check all boxes

Install	Package	Directory	Comment
☐	Developer Support 5.4.3b1	/packages/acs-developer-support	New install.
☐	Bookmark 1.0	/packages/bookmark	New install.
☐	dotLRN LORS Management Applet 2.4.1	/packages/dotlrn-lorsm	New install.
☐	dotLRN Random Photo 2.4.1	/packages/dotlrn-random-photo	New install.

Figura A.6: Página de instalación de paquetes 1

proyecto/instalacion7.jpg

☐	Random Photo Portlet 2.4.1	/packages/random-photo-portlet	New install.
☑	Rating 1.0	/packages/rating	New install.
☐	Selva Theme 2.4.1	/packages/theme-selva	New install.
☐	Trackback 0.1	/packages/trackback	New install.
☐	Tsearch2 Driver 5.4.3	/packages/tsearch2-driver	New install.
☐	Weblogger Portlet 2.4.1	/packages/weblogger-portlet	New install.
☐	XML-RPC Server 0.3	/packages/xml-rpc	New install.

Next -->

Figura A.7: Página de instalación de paquetes 2

proyecto/instalacion8.jpg

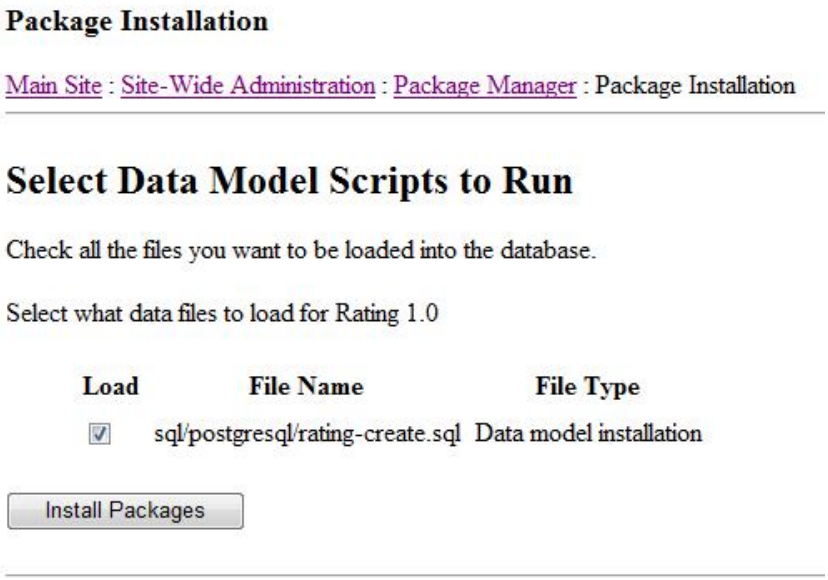

Figura A.8: Página de instalación de paquete Rating

proyecto/instalacion9.jpg

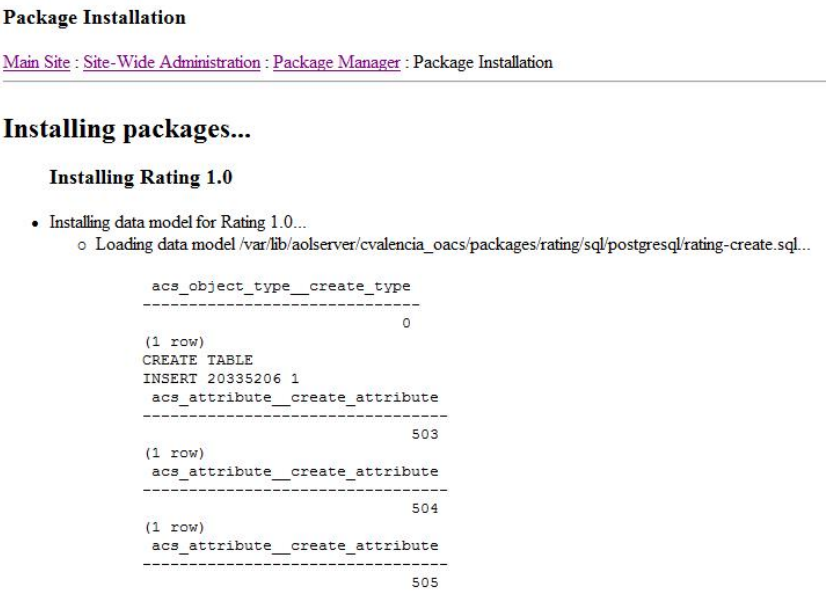

Figura A.9: Proceso de instalación de Rating 1

proyecto/instalacion10.jpg

```
(1 row)
CREATE TABLE
INSERT 20335251 1
 acs_attribute__create_attribute
---------------------------------
                             509
(1 row)
 acs_attribute__create_attribute
---------------------------------
                             510
(1 row)
 acs_attribute__create_attribute
---------------------------------
                             511
(1 row)
```

Installed Rating, version 1.0.

Package enabled.

Mounted an instance of the package at /rating

Done installing packages.

You should restart the server now to make installed and upgraded packages available. Click here to restart the server now.

Figura A.10: Página de instalación de Rating 2

proyecto/instalacion11.jpg

Package Installation

Main Site : Site-Wide Administration : Package Manager : Package Installation

Please wait while the installer searches your system for packages to install ...

Done.

Select Packages to Install

Please select the set of packages you'd like to install.

uncheck all boxes | check all boxes

Install	Package	Directory	Comment
☐	Developer Support 5.4.3b1	/packages/acs-developer-support	New install.
☑	Bookmark 1.0	/packages/bookmark	New install.
☐	dotLRN LORS Management Applet 2.4.1	/packages/dotlrn-lorsm	New install.

Figura A.11: Página de instalación de paquetes 3

proyecto/instalacion12.jpg

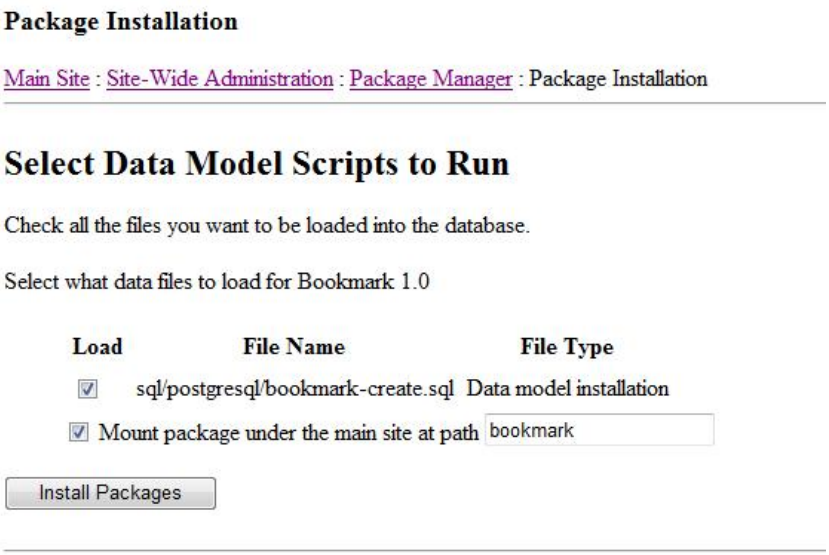

Figura A.12: Página de instalación de paquete Bookmark

proyecto/instalacion13.jpg

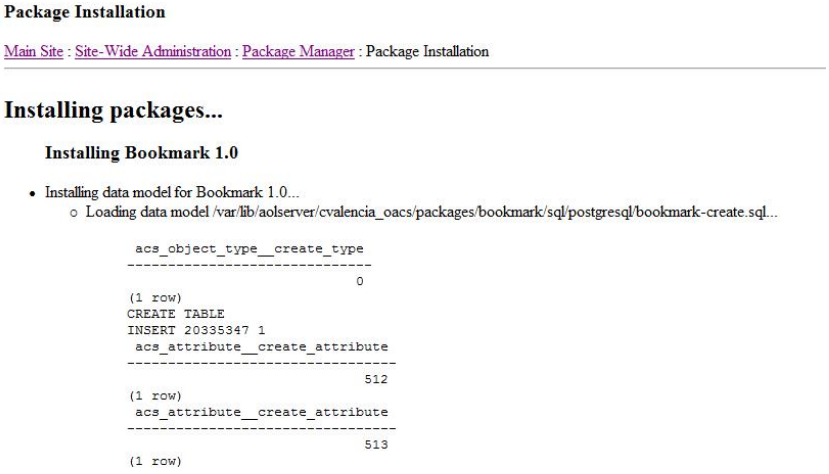

Figura A.13: Proceso de instalación de Bookmark 1

proyecto/instalacion14.jpg

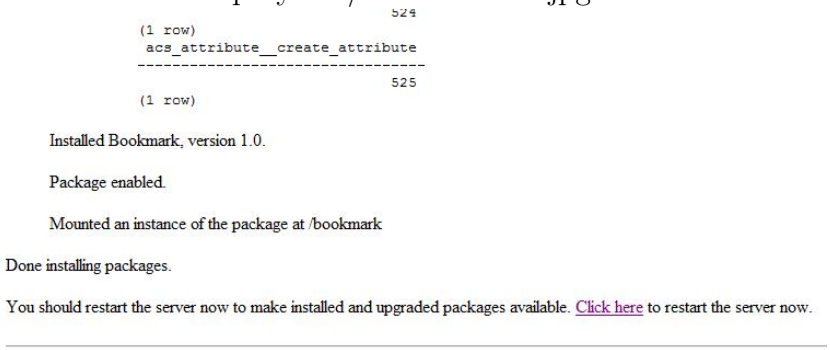

Figura A.14: Proceso de instalación de Bookmark 2

Apéndice B

Manual de usuario

El diseño de la interfaz de usuario de la aplicación es bastante sencillo e intuitivo ya que este fue uno de los objetivos que se buscaban de modo que incluir esta aplicación en una comunidad de usuarios no necesitase un periodo de aprendizaje. En este apéndice se comentarán los aspectos principales del uso de esta aplicación.

La primera vez que un usuario acceda, le aparece una página como la de la figura B.1 en la que se pide indicar si tiene usuario en delicuous y en caso afirmativo, introducir el nombre de usuario en delicious. Si todavía no usa delicious o no desea dar esa información en ese momento, se puede añadir esta información posteriormente en la página de configuración de usuario de la aplicación.

Una vez inscrito se muestra la página principal de la aplicación, punto de entrada cada vez que accedamos al servicio de gestión de *bookmarks*. En esta página encontramos el listado de *bookmarks* de la comunidad. La primera vez que entremos puede estar vacía si no se han agregado *bookmarks* hasta ese momento porque todavía no se ha ejecutado la adquisición de *bookmarks* de delicious ni se han agregado manualmente o con contenido como en la figura B.2. En esta página hay dos botones y una tabla. Los botones se utilizan para ir a la página en la que se crear *bookmarks* manualmente y para acceder a la configuración de usuario de la aplicación. La tabla muestra los resultados que cumplen con las preferencias de cada usuario.

Cuando se pulsa el botón de añadir *bookmark* se abre la página mostrada en la figura B.3. Después de rellenar los campos obligatorios se puede enviar la información pulsando el botón OK y se crea un nuevo bookmark si este no existía previamente (figura B.4) o devuelve un mensaje de error indicando que el *bookmark* ya existe (figura B.5).

Desde la página principal se puede acceder a la página de configuración de usuario (figura B.6) pulsando el botón Configuración. En esta página se puede

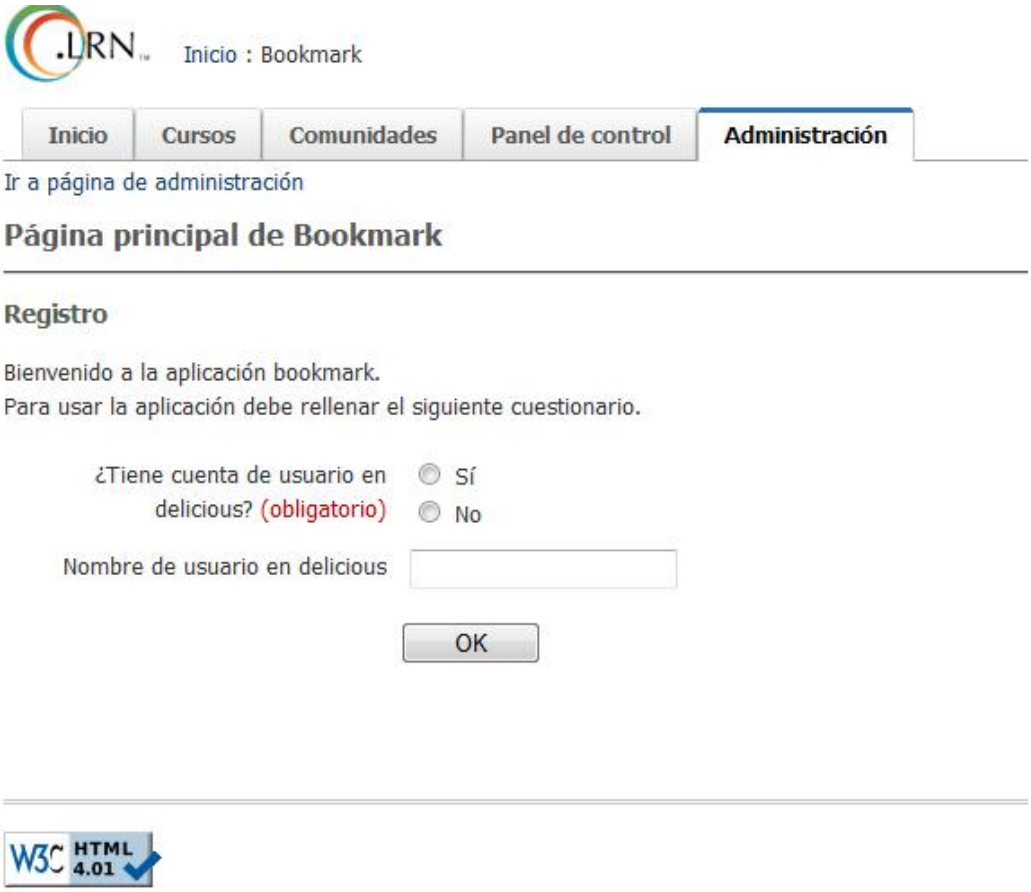

Figura B.1: Página inscripción aplicación

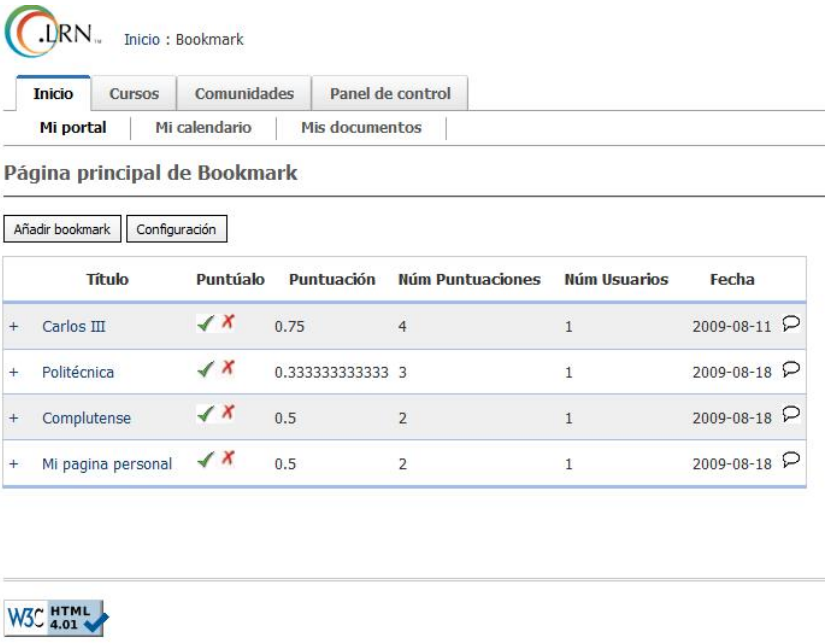

Figura B.2: Página principal de Bookmark

modificar el nombre de usuario de cuenta de delicious así como cambiar las preferencias de visualización pudiendo utilizar un filtrado adicional eligiendo una de las etiquetas de todos los elementos almacenados y cambiando el orden (por antigüedad o por valoración) en el que se muestran.

En la tabla de *bookmarks* que aparece en la figura B.2 se puede ver como en cada elemento de la lista aparece un signo + a la izquierda. Pulsando ese signo se abre una página como la de la figura B.7 en la que se ven todos los datos de un determinado *bookmark*. Pulsando sobre el título del *bookmark* nos enlaza directamente con la página a la que se refiere dicha fila de la tabla. La columna puntúalo es donde se indica si el *bookmark* es útil o no, después aparecen los datos referidos a la evaluación de todos los usuarios de dicho elemento y por último aparece la fecha en la que se adquirió el *bookmark* en la cuenta de delicious o el día que se agregó manualmente a la aplicación y un icono que abre la página para realizar comentarios sobre el *bookmark*.

Por último queda comentar como se utilizan los comentarios y las descripciones. La página de comentarios (figura B.8) es una página ofrecida para mantener conversaciones tanto síncronas como asíncronas entre los usuarios de la aplicación. Para ello, cada vez que un usuario envía un texto nuevo se almacena como un nuevo comentario que se muestran en el orden en el que se han escrito.

La página de descripciones (figura B.9) esta diseñada para ofrecer un

Figura B.3: Página para añadir bookmarks manualmente

Figura B.4: Bookmark añadido correctamente

Figura B.5: Error al añadir bookmark manualmente

Figura B.6: Página de configuración de usuario

Figura B.7: Página de información sobre un bookmark

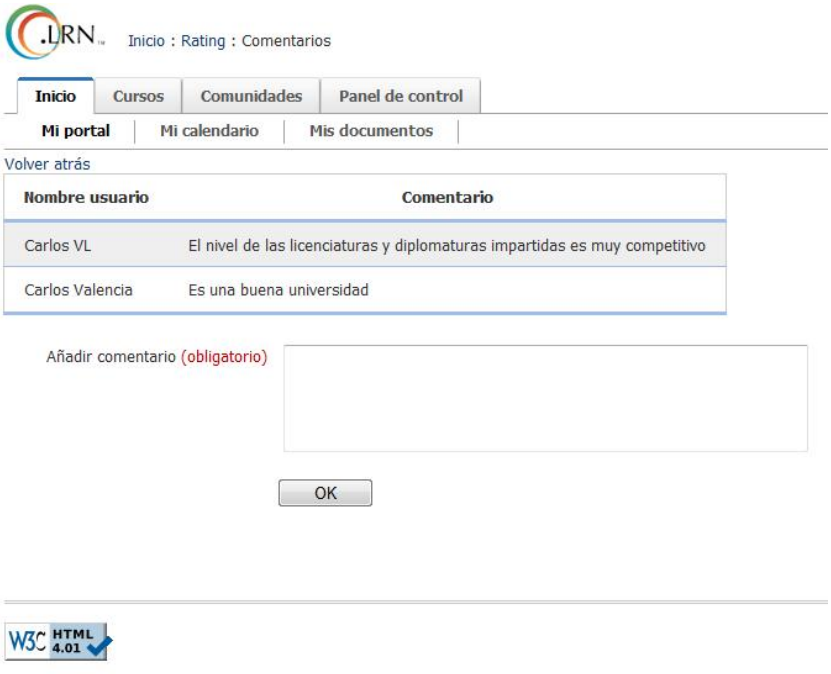

Figura B.8: Página de comentarios

espacio a los usuarios para dar una explicación del contenido de la página enlazada o la utilidad de la misma. Cada usuario dispone de un bloque de texto que puede editar después de haberlo escrito una vez. Por defecto se incluye la descripción obtenida de delicious para dicho usuario y lo que hace más interesante esta página de descripciones es el sistema de ordenación que tiene en función de las votaciones de los usuarios de modo que las descripciones mejor valoradas se muestran en los primeros puestos.

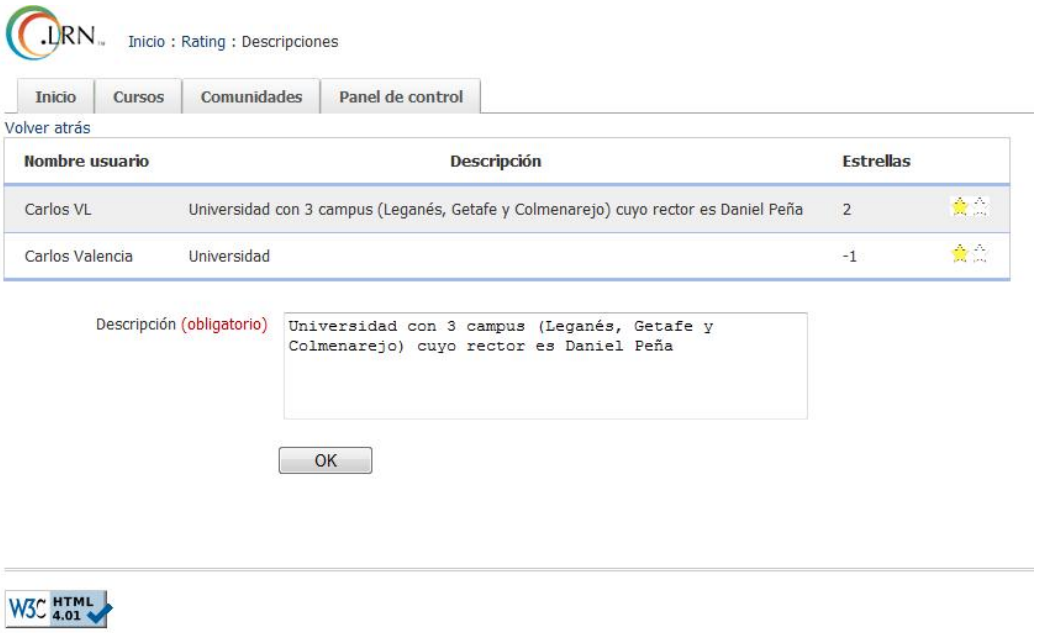

Figura B.9: Página de descripciones

B.1. Perfil administrador

Los usuarios con perfil de administración pueden acceder a los mismos contenidos que los demás usuarios pero además tienen el control del comportamiento de la aplicación. Para acceder a la página de configuración del servicio se debe pulsar el enlace a Ír a página de administración'que aparece en la parte superior de la pantalla principal. Podemos ver este enlace en la figura B.10

Al pulsar en el enlace nos aparece la página de administración de la aplicación (figura B.11). En ella lo primero que encontramos es un enlace que dice Éjecutar actualización de delicious'con la que se realiza una actualización de la base de datos adquiriendo la nueva información procedente de delicious.

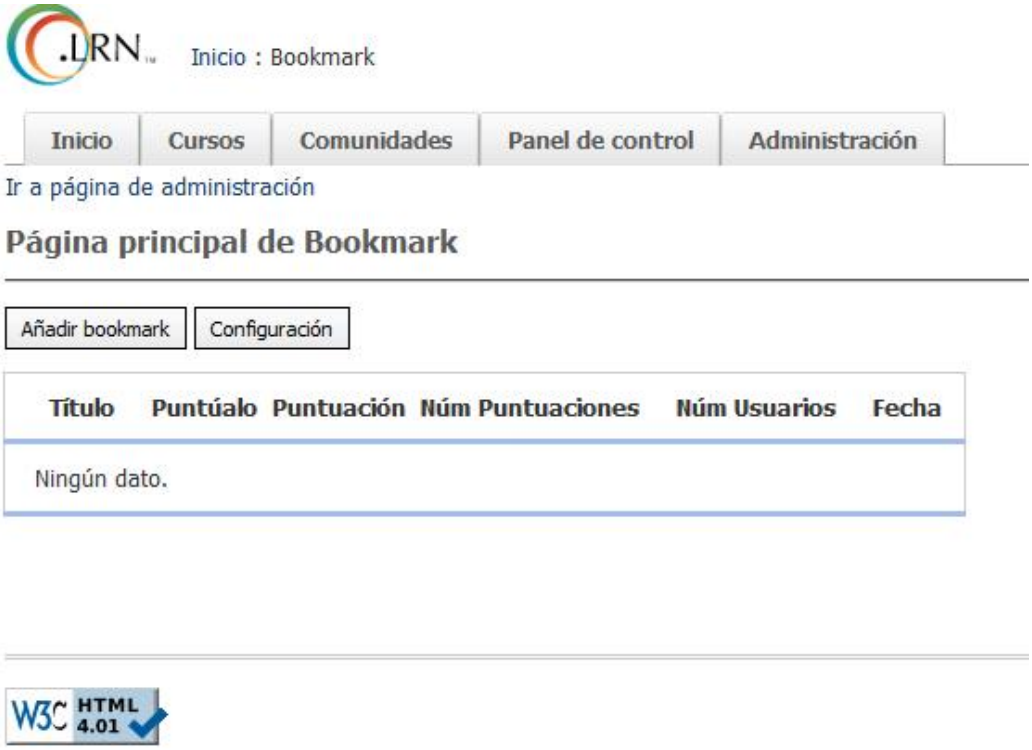

Figura B.10: Página principal vacía de usuario con privilegios

La aplicación realiza este proceso periódicamente de forma automática pero se ofrece esta posibilidad en caso de querer agregar información de interés en el momento y sin necesidad de añadir los nuevos *bookmarks* de forma manual.

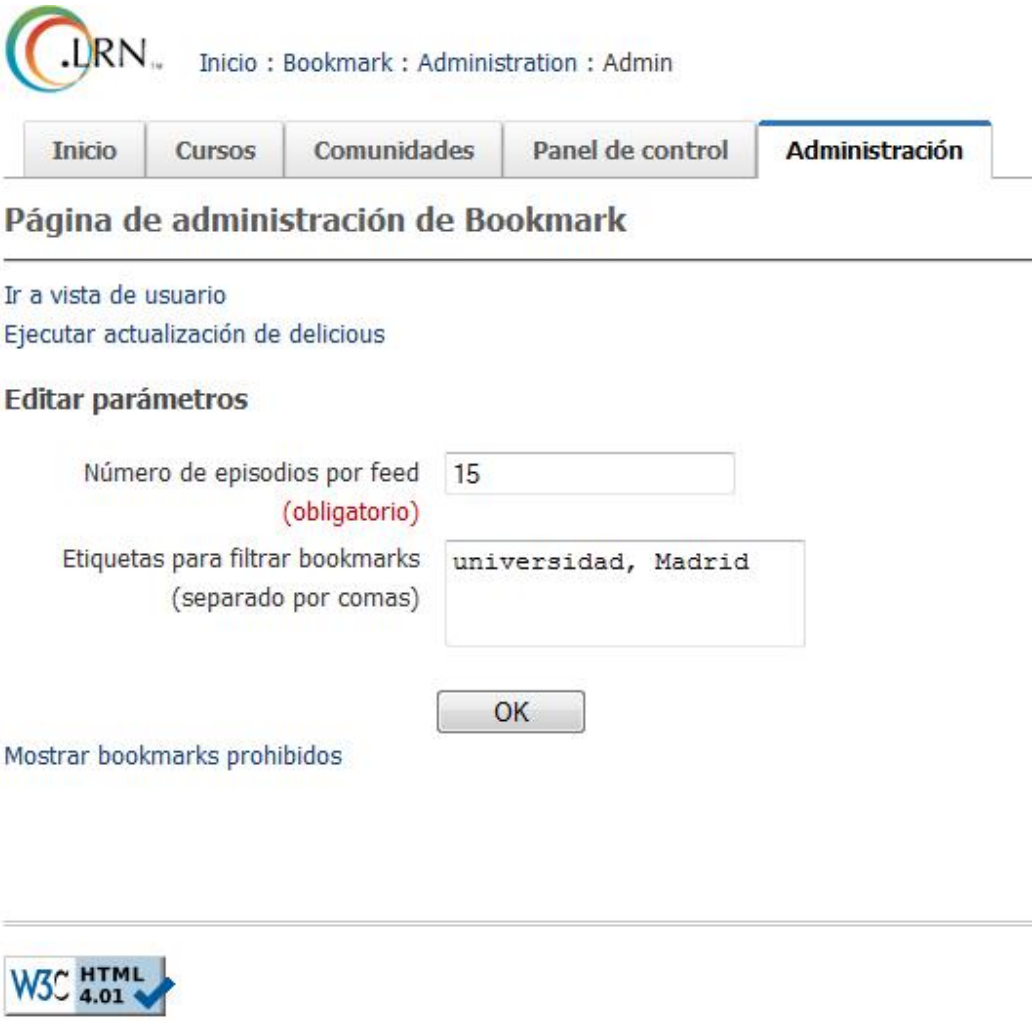

Figura B.11: Página administración de Bookmark

Después aparece una sección para editar los parámetros de configuración de la aplicación. El parámetro número de episodios por feed sirve para indicar cuantos *bookmarks* queremos pedir por cada usuario de delicious. Con este parámetro podemos controlar la cantidad de información que almacenamos ya que si el número de usuarios es grande puede que interese obtener menos episodios de cada usuario de modo que tengamos menos *bookmarks* almacenados. El parámetro etiquetas para filtrar *bookmarks* nos ofrece la posibilidad de realizar un primer proceso de filtrado de la información que obtenemos

de delicious de modo que sólo obtendremos episodios de *bookmarks* que contengan las etiquetas seleccionadas.

En esta página también encontramos otro enlace, 'Mostrar bookmarks prohibidos'. Dicho enlace despliega en la misma página una lista de los *bookmarks* que han sido prohibidos por los administradores de la aplicación (ver figura B.12). Estos pueden haber sido prohibidos por diversos motivos ya sea por contenidos inapropiados, spam... Los enlaces prohibidos se muestran junto con un comentario escrito por el administrador que prohibió el *bookmark* explicando por qué razón se prohibió. En esta sección de la página también aparece un botón que nos redirige a la página desde la cuál se prohiben *bookmarks* existentes.

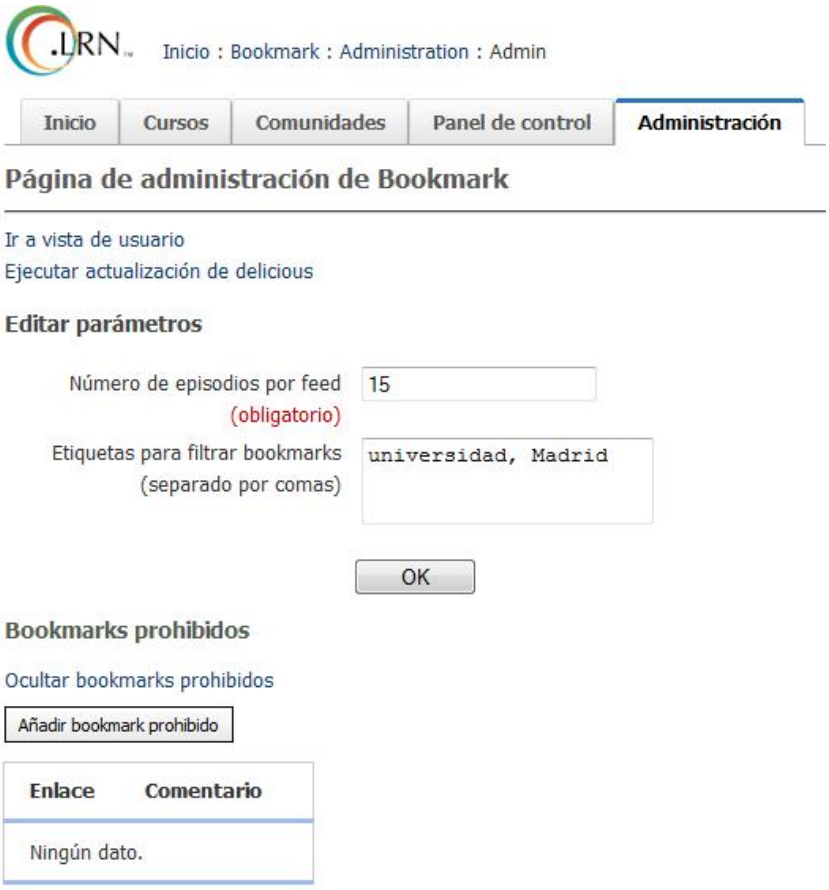

Figura B.12: Página de administración de Bookmark mostrando bookmarks prohibidos

En la figura B.13 se ve la página desde la cuál se prohiben los *bookmarks* con contenido inapropiado. En esta página se muestra una lista con todos

los *bookmarks* que son visibles. Haciendo click en las cajas que aparecen a la izquierda de cada elemento podemos seleccionarlo para después prohibirlos pulsando el botón 'prohibir bookmarks'. Por defecto, se almacena como comentario del *bookmark* prohibido la descripción que se almacenó del mismo, pero se puede modificar para guardar información más útil sobre las razones por las que se bloqueó o cualquier otra información que también pueda ser de utilidad.

Figura B.13: Página para agregar bookmarks prohibidos

En el ejemplo se puede ver como Mi página personal no es una universidad por lo que se prohibe seleccionando el *bookmark* correspondiente y pulsando en el botón Prohibir bookmarks. Hecho esto la aplicación se redirige a la página de administración y allí se puede ver en la lista de *bookmarks* prohibidos el que se acaba de eliminar (figura B.14). Este *bookmark* ya no aparecerá en

la página principal del servicio y tampoco se podrá añadir como un nuevo elemento. En la figura B.14 se puede ver que a la derecha del elemento que aparece en la tabla aparecen dos iconos. El primero sirve para editar el comentario que tiene cada elemento y el botón de la derecha para borrar el elemento.

Figura B.14: Lista de bookmarks prohibidos con un elemento

Apéndice C

Enlaces de servicios y plataformas mencionados en la memoria

A continuación se presentan las páginas oficiales de cada uno de los servicios y plataformas mencionados en la memoria. En el siguiente listado se ofrece la URL de la página oficial, pero también la frase o slogan que representa a dicha herramienta. Para más información de cualquiera de las herramientas entre en su sitio oficial.

- .LRN

  .LRN is the world's most widely adopted enterprise-class open source software for supporting e-learning and digital communities.

  `www.dotlrn.com`

- OpenACS

  OpenACS is a toolkit for building scalable, community-oriented web applications. OpenACS is the foundation for many products and websites, including the .LRN (pronounced d̈ot learn) e-learning platform.

  `www.openacs.org`

- AOLserver

  AOLserver is America Online's Open-Source web server. AOLserver is the backbone of the largest and busiest production environments in the world. AOLserver is a multithreaded, Tcl-enabled web server used for large scale, dynamic web sites.

  `www.aolserver.com`

- TCL

  TCL Developer Xchange

  `www.tcl.tk`

- PostgreSQL

  The world's most advanced open source database.

  `www.postgresql.org`

- Interbase

  Ensure durability and reliability leveraging the powerful database for embedded and enterprise applications

  `www.codegear.com/products/interbase`

- MySQL

  The world's most popular open source database.

  `www.mysql.com`

- Oracle

  Base de datos más usada del mundo

  `www.oracle.com/database/index.html`

- Oracle

  .LRN virtual machine for VM-WARE

  `https://gradient.it.uc3m.es/xowiki/dotlrn_vm`

- RSS

  Really Simple Syndication specifications, tutorials and discussion

  `www.rssboard.org`

- SVN

  Open source control version system

  `subversion.tigris.org`

- Flickr

  Servicio de compartición de fotografías.

  `www.flickr.com`

- scholar.com

  Servicio web que crea redes sociales

  `www.scholar.com`

Servicios de gestión de *bookmarks* mencionados en la memoria:

- Delicious

  The tastiest bookmarks on the web.

  `delicious.com`

- Blinklist

  Easily Save Your Links for Later

  `www.blinklist.com`

- Stumbleupon

  To discover the best of the web

  `www.stumbleupon.com`

- Connotea

  Organise. Share. Discover

  `www.connotea.org`

- CiteUlike

  Free service for managing and discovering scholarly references

  `www.citeulike.org`

- Diigo

  Highlight and Share the web!

  `www.diigo.com`

- ifavoritos

  Servicio de gestión de marcadores en español

  `www.ifavoritos.com`

- favoriting

  Servicio de gestión de marcadores en español

  `www.favoriting.com`

- Mister Wong

  Servicio de gestión de marcadores en español

  `www.mister-wong.es`

Bibliografía

[1] Administrator's guide, 2008. http://www.openacs.org/doc/acs-admin.html.

[2] For openacs package developers, 2008. http://www.openacs.org/doc/acs-package-dev.html.

[3] Forum of openacs community, 2008. http://www.openacs.org/forums/.

[4] Openacs tutorial by jade rubick, 2008. http://www.rubick.com/openacs/.

[5] Acerca de moodle, 2009. http://docs.moodle.org/es/Acerca_de_Moodle.

[6] Claroline.net let's bould knowledge together, 2009. http://www.claroline.net.

[7] Delicious, social bookmarking, 2009. http://delicious.com/.

[8] .lrn is the world's most adopted enterprise-class open source software for supporting e-learning and digital communities., 2009. http://dotlrn.org/.

[9] Tcl history, 2009. http://www.tcl.tk/about/history.html.

[10] Clif Flynt. *Tcl/Tk: a developer's guide*. Morgan Kaufmann, 2003.

[11] Jose Jesús García Rueda. Material sobre pruebas en software de laboratorio de programación, 2004.

[12] Ben Hammersley. *Developing Feeds with RSS and Atom*. O'Reilly, 2005.

[13] Brett O'Connor. *Begining del.icio.us Mashups*. WROX Press, 2007.

[14] Simon Pickin. Material de software de comunicaciones, 2007.

[15] E. Pérez. Plataforma para el diseño y la realización de exámenes en el entorno openacs/.lrn. Master's thesis, Universidad Carlos III, Madrid, 2000.

[16] Paul Raines. *TCL/TK in a nutshell: a desktop quick reference*. O'Reilly, 1999.

[17] Óscar Hernando Guzmán Cortes. Aplicación práctica del diseño de pruebas de software a nivel de programación. 2004.

[18] Joshua D. Worsley, John C.; Drake. *Practical PostgreSQL*. O'Reilly, 2002.